П.В. Сенченко

НАДЕЖНОСТЬ, ЭРГОНОМИКА И КАЧЕСТВО АСОИУ

Учебное пособие

Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

П.В. Сенченко

НАДЕЖНОСТЬ, ЭРГОНОМИКА И КАЧЕСТВО АСОИУ

Учебное пособие

Томск 2016

Сенченко П.В.

Надежность, эргономика и качество АСОИУ: Учебное пособие. – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2016. – 189 с.

В учебном пособии рассматриваются вопросы надежности, эргономики и качества программного обеспечения автоматизированных систем обработки информации и управления. Изложены основные элементы теории надежности, рассматриваются стандарты качества. Особое внимание уделяется различным методам повышения надежности и качества создаваемых систем, введению различного рода избыточности, организационным мероприятиям, позволяющим улучшить показатели надежности и качества АСОИУ. Даются рекомендации по созданию эргономичных АСОИУ. Материал подготовлен на основе учебного курса, который читается автором в Томском государственном университете систем управления и радиоэлектроники.

Учебное пособие ориентировано на студентов направлений подготовки бакалавров «Программная инженерия», «Бизнес-информатика» а также студентов родственных специальностей.

Содержание

Никогда не выявляйте в программе ошибки, если не знаете, что с ними делать дальше.

Руководство по системному программированию Штейнбаха

ВВЕДЕНИЕ

В настоящее время при разработке автоматизированных информационных систем (АИС) любого класса сложности все больше внимания уделяется проверке качества создаваемого продукта. Стремление создавать системы, удовлетворяющие действующим стандартам, позволяет достичь требуемого уровня качества таких систем. При этом отдельное место занимают проблемы обеспечения эргономичности создаваемых программных продуктов.

В пособии рассматриваются проблемы обеспечения качества отдельных элементов и сложных информационных систем в целом, влияние программного обеспечения на безаварийную работу систем. Особое внимание при изложении материала уделяется различным методам повышения качества создаваемых систем, введению различного рода избыточности, организационным мероприятиям, позволяющим улучшить показатели качества автоматизированных информационных систем, описываются различные типы тестирования информационных систем. Рассматриваются основные стандарты качества автоматизированных систем обработки информации и управления (АСОИУ). Отдельный раздел курса посвящен эргономическим характеристикам АСОИУ, даются определения понятия эргономики, затрагиваются проблемы обеспечения эргономического качества систем.

Изучение материала, представленного в пособии, предполагает знание студентами таких дисциплин, как «Метрология, стандартизация и сертификация», «Информационные технологии», «Интерфейсы АСОИУ».

1. ПРИНЦИПЫ ОРГАНИЗАЦИИ РАЗРАБОТКИ АСОИУ[1]

1.1. Краткая характеристика АСОИУ

АСОИУ описывается как **система**, представляющая собой совокупность средств сбора, хранения и отображения информации, аппаратных, математических и телекоммуникационных средств. В большинстве своем АСОИУ относятся к классу систем «человек-компьютер» (человеко-машинные системы). АСОИУ находят применение во всех областях жизнедеятельности человека, к ним можно отнести и системы учета населения, и бухгалтерские комплексы программ и даже системы управления атомными электростанциями. Независимо от области применения **основными функциями систем** являются обработка некоторой информации и управление различными процессами.

Сложная система рассматривается как совокупность взаимосвязанных подсистем, ориентированных на достижение единых целей. АСОИУ, в свою очередь, может представлять собой как отдельную ***монофункциональную систему***, так и сложную ***многофункциональную систему*** (например, система персонального учета населения), состоящую из набора взаимосвязанных, использующих единую информационную среду подсистем. Естественно, что от области применения АСОИУ зависит основная функция системы, которая определяет соответственно набор функциональных подсистем, при этом в состав АСОИУ могут быть включены некие вспомогательные подсистемы (подсистемы пользовательского аудита, разграничения полномочий доступа и защиты данных и др.). Кроме этого в состав АСОИУ могут быть включены аппаратные средства, характерные для конкретной области применения. В настоящее время состав подсистем АСОИУ не регламентирован какими-либо стандартами и зависит от назначения и конкретной реализации системы.

Проектирование АСОИУ подразумевает разработку различных уровней моделей системы. Для функционального, ин-

[1] Данная глава написана по материалам, представленным в [1, 2].

формационного и имитационного моделировании будущих АСОИУ возможно использование различных методологий. Например, функциональная модель системы может быть реализована виде SADT-диаграмм (Structured Analysis and Design Technique — технология структурного анализа и моделирования). Техническими средствами реализации функциональных моделей являются пакеты BPwin, IDEF/Design, Visio Professional и др. (обзор средств проектирования и моделирования систем представлен в п. 1.3.2).

1.2. Стадии разработки системы

1.2.1. Обзор стадий разработки системы

Процесс разработки АСОИУ состоит из следующих стадий (этапов):

1) планирование;
2) проектирование;
3) кодирование;
4) тестирование;
5) документирование;
6) внедрение;
7) сопровождение.

Обычно этапы жизненного цикла создания АИС описывают последовательно строго друг за другом — заканчивается один и начинается другой, однако в действительности этапы жизненного цикла в значительной степени перекрываются. При этом на каждом из этапов обязательно возникают ошибки, и стоимость их исправления будет различной на каждом этапе.

На этапе разработки технических требований стоимость внесения изменений в проектную документацию относительно невысока. Но после написания программного кода ситуация резко меняется: любое изменение проектной документации влечет за собой затраты и на переработку кода, часто гораздо более значительную, чем может показаться при оценке количества изменений в спецификации.

Исправление программистом ошибок, обнаруженных в ходе написания системы, не влечет больших затрат. Ему не нужно

взаимодействовать с другими сотрудниками, не нужно ничего никому объяснять. Ему незачем вносить описание ошибок в общую базу данных, служащую для контроля их исправления, а тестировщикам и руководителю не нужно осуществлять этот контроль. Такая ошибка никого не задерживает и не мешает ничьей работе.

В случае внесения исправлений в разрабатываемую версию системы ошибку исправить гораздо дешевле, чем после внедрения системы, когда каждое исправление или новую версию АИС приходиться высылать каждому пользователю либо осуществлять непосредственный выезд сотрудника компании к заказчику.

Следует учесть, что чем позднее обнаруживается ошибка, тем дороже обходится ее исправление. Стоимость исправления ошибок растет экспоненциально: легче всего это делать на стадии планирования, а по мере перехода к проектированию, кодированию, тестированию и сопровождению она значительно увеличивается. Таким образом, ***чем раньше найти и исправить ошибку, тем дешевле это обойдется компании-разработчику.***

В данном разделе подробно рассмотрим стадии планирования и проектирования в соответствии с действующей системой стандартов. Характерной особенностью предложенной интерпретации этапов разработки АСОИУ является наличие практически на каждом из этапов создания систем элементов тестирования, являющих собой мощный инструмент обеспечения требуемого уровня надежности и качества АСОИУ. Этим вопросам посвящен шестой раздел пособия.

На стадии документирования разрабатывается техническая документация на систему (см. раздел 5). Помимо собственно документации на этой стадии разрабатывается также справочная система, от качественной реализации которой напрямую зависит адекватность восприятия конечным пользователем приобретаемой информационной системы.

1.2.2. Стадия планирования

Стадия планирования включает следующие этапы:

- определение целей;

- анализ требований;
- определение функциональных характеристик продукта;
- тестирование на этапе планирования.

Определение целей

Планирование начинается с формирования общего видения конечного программного продукта. Составляемый при этом документ не отличается детальностью. В нем может быть описан пользовательский интерфейс, требования к надежности программного продукта и его производительности. В этом же документе определяется предполагаемая стоимость продукта и затраты на его разработку. Он разрабатывается, прежде всего, для постановки перед командой разработчиков конкретной цели. При этом конечный результат может не соответствовать первоначальному описанию.

Анализ требований

Требования к программному продукту, вырабатываемые на данном этапе, подлежат обязательному выполнению. Они носят функциональный характер, а за практическую реализацию отвечают разработчики. Перечень требований может охватывать стоимость будущего продукта, его производительность, надежность, а также некоторые элементы пользовательского интерфейса. Степень подробности при описании требований зависит от конкретной специфики разработки. Спецификация (определение требований к программе) — один из важнейших этапов, на котором подробно описывается исходная информация; формулируются требования к результату; определяется поведение программы в особых случаях (например, при вводе неверных данных); разрабатываются диалоговые окна, обеспечивающие взаимодействие пользователя и программы.

В требования к программному продукту или другие документы, вырабатываемые на ранних стадиях планирования, включаются также и требования к аппаратному обеспечению. Чтобы не усложнять материал, в этом разделе мы не будем касаться таких специфических вопросов, как одновременная разработка аппаратного и программного обеспечения или периодическое повышение требований к аппаратному обеспечению в соответ-

ствии с его развитием. Будем исходить из предположения, что типы аппаратных средств, на которые ориентирован программный продукт, известны с самого начала разработки.

Определение функциональных характеристик программного продукта

Этап определения функциональных характеристик можно представить как связующее звено между разработкой требований к программному продукту и будущим инженерно-проектными документам. Определение требований к программному продукту напрямую связаны с маркетинговой деятельностью компании. При этом разработчику необходимо нечто более определенное, полное и конкретное. Это «нечто» и есть функциональные характеристики, среди которых представлен перечень функций будущей информационной системы, описание входных и выходных документов. Способов реализаций этих функций данный документ касается, только в самых необходимых случаях, когда это позволяет получить более полное представление о документе. В таком случае в документе в общих чертах описывается возможная реализация, но конечная внутренняя и внешняя структура продукта, может оказаться иной.

Существует два принципиально разных подхода к определению функциональности системы. При первом подходе система снабжается максимальным количеством функций, результаты многих из которых являются суммой результатов других функций. При втором подходе все составные функции (метафункции) из системы изымаются.

Оба подхода имеют как недостатки, так и достоинства. Подход, при котором количество функций ограничено, позволяет упрощать интерфейс, но при этом пользователю необходимо понимать, как из многих низкоуровневых функций «собирать» функции более сложные. Подход, при котором помимо низкоуровневых функций есть высокоуровневые, позволяет потенциально обеспечивать большую скорость работы за счет отсутствия пауз между низкоуровневыми функциями, но при этом требует от пользователя знаний о местоположении высокоуровневых функций. Однако высокоуровневые функции способны значительно перегружать пользовательский интерфейс.

Тестирование на этапе планирования

На этом этапе тестируются не программы — «тестируются» идеи. К их анализу привлекаются специалисты по маркетингу, руководители проекта, главные конструкторы и специалисты по анализу человеческого фактора. А вот члены группы тестирования участвуют в этой работе достаточно редко.

Суть тестирования на этом этапе заключается в следующем. Группа аналитиков читает черновики проектных документов, затем собирается информация, необходимая для их оценки и дальнейшего планирования. Для этого существует несколько стандартных способов: сравнительный анализ, дискуссионные группы и обследование объекта. Результаты каждой из этих процедур могут привести к значительному пересмотру существующих планов.

При анализе и оценке требований к продукту и его функциональным характеристикам специалисты, прежде всего, пытаются выяснить следующее.

Адекватны ли эти требования? Действительно ли именно такую информационную систему хочет создать компания?

Полны ли вырабатываемые требования? Не упущены ли какие-либо полезные или даже жизненно необходимые функции? Нельзя ли ослабить какие-либо из перечисленных требований?

Совместимы ли требования между собой? Требования к системе (и ее функции) могут оказаться логически или психологически несовместимыми. Логическая несовместимость означает их противоречивость, а психологическая — концептуальные разногласия (разобравшись с одной из функций, пользователь может не понять другую).

Выполнимы ли требования? Не требуется ли для нормальной эксплуатации продукта более мощное, чем указано в документации, аппаратное обеспечение (больший объем памяти, более высокая пропускная способность, большее разрешение и т. д.)?

Разумны ли вырабатываемые требования к создаваемому продукту? Время и соответственно стоимость создания системы прямо пропорционально увеличиваются при расширении требований к системе. С одной стороны — необходимо обеспечить оптимальную производительность продукта, его надежность и

нетребовательность к ресурсам, а с другой — минимизировать время и стоимость его разработки. Таким образом, необходимо определить оптимальное соответствие между этими характеристиками. Поэтому одним из ключевых вопросов планирования является правильная расстановка приоритетов реализуемых в системе и исключение из списка необоснованных требований.

Поддаются ли вырабатываемые требования тестированию? Насколько легко можно будет определить, соответствует ли инженерно-проектная документация требованиям к программному продукту.

На практике возникают ситуации, когда специалисты на стадии планирования игнорируют проблемы совместимости функций и сложности программного продукта. При этом составляется список удачных идей конкурентов. Сам по себе этот список может быть очень полезным, но если рассматривать его как требования к продукту, его необходимо значительно сократить. Отобрать самое существенное можно, используя два следующих метода: метод дискуссий и обследование объекта.

Работа дискуссионных групп

Обычно каждая создаваемая информационная система предназначается для определенного сегмента рынка. Целью данного метода анализа является определение ключевых требований этого сегмента. Для этого аналитик отбирает небольшую группу людей, являющихся, по его мнению, наиболее типичными представителями нужного сегмента рынка. Члены группы не знают друг друга. Аналитик предлагает им обсудить определенную тему. Предлагаемых к обсуждению тем не должно быть слишком много. Аналитик может направлять дискуссию, задавая наводящие вопросы и концентрируя внимание группы на заданной теме, а может и не участвовать в обсуждении. Его цель — выяснить реакцию рынка на предложенную идею, при этом ни в чем не убеждая членов группы.

Такая дискуссия может осветить самые различные аспекты обсуждаемой проблемы. Аналитик может понять, чего ждут пользователи от данного типа продуктов, как они намерены их использовать, какие функции для них наиболее важны. Можно сконцентрировать группу и на одной конкретной функции про-

дукта или на одной цели его применения. Можно использовать группу для генерации идей еще до детального планирования, а можно проанализировать ее реакцию на элементы уже готового плана.

Обследование объекта

Каждая информационная система предназначается для полной или частичной автоматизации выполнения некоторой задачи, возможно, очень сложной. Чтобы как можно четче представить себе эту задачу, аналитик выполняет обследование объекта автоматизации. Он наблюдает людей за работой, беседует с ними, пытается выявить все, в чем продукт может помочь своим будущим пользователям. Аналитик спрашивает себя, в чем же конкретно состоит изучаемая им задача. Как люди выполняют ее без проектируемого продукта? Какова последовательность их действий? Почему она именно такая? В какой момент работнику нужна каждая конкретная информация и для чего? Что особенно сильно замедляет их работу, и почему эта проблема не решена до сих пор? Работа специалиста по обследованию является частью проектирования продукта и жизненно необходима для разработки как пользовательского интерфейса, так и внутренней структуры системы.

Обследование объекта может быть выполнено и после определения требований к продукту. По результатам обследования эти требования могут быть сильно изменены. На этом этапе может быть полезна методика сравнения систем, предложенная в подразделе 2.5.

1.2.3. Стадия проектирования

На этапе проектирования группа соответствующих специалистов решает, как будут реализованы запланированные возможности системы. Разрабатывается ***внешний дизайн*** программного продукта (вид продукта с точки зрения пользователя) и его ***внутренняя структура.*** Эти составляющие тесно взаимосвязаны и проектируются одновременно. Разрабатывая проект, специалисты опираются на требования к системе, если этого документа нет либо он неполон или постоянно меняется, им приходится планировать функции продукта самостоятельно.

По классической модели разработки программного обеспечения кодирование начинается только по завершении этапа проектирования. Однако это не касается прототипа, создаваемого в рамках проектирования для анализа и демонстрации будущего продукта. На практике довольно значительная часть кода прототипа может оказаться в конечном продукте. На этапе проектирования могут быть написаны и некоторые низкоуровневые процедуры, к которым предъявляются наиболее строгие требования по скорости и потреблению ресурсов.

Описание внутренней структуры программного продукта определяет набор будущих программных модулей **(*программную архитектуру*),** структуру, взаимосвязи и принципы хранения и обработки данных (***организацию данных***) и ***алгоритмы*** работы программы.

Проектирование программной архитектуры

На данном этапе проводится декомпозиция каждой задачи до выделения достаточно самостоятельных элементов, которые можно реализовать в виде отдельных программных модулей или процедур. Обычно сложные программные продукты реализуются в виде ***системы*** — набора связанных между собой полноценных программ. Такие программы часто называют ***процессами***, особенно если они работают параллельно. Хотя процессы могут работать и независимо, как правило, они взаимодействуют между собой. Например, они могут использовать одни и те же данные, или один из них может выполнять определенные задания по запросу другого.

Документация, определяющая принципы и правила взаимодействия процессов системы, называется *протоколом.* В описании программной архитектуры системы определяются ее основные компоненты и использующиеся для их взаимодействия коммуникационные протоколы.

Как и любые другие программы, процессы поддаются декомпозиции. Разбиение программы на модули называют модульной декомпозицией. Под модулем в данном случае понимают часть кода, которая может рассматриваться как независимое целое и имеет единственную точку входа. В терминологии программиста этому определению соответствуют процедуры и функ-

ции. Обычно модуль реализует либо одно конкретное задание, либо четко определенную группу заданий, для выполнения которых другие модули могут его вызывать. Вызывающий модуль передает вызываемому для обработки некоторые данные, а тот, в свою очередь, возвращает результат.

Проектирование данных

Разработчик структуры данных должен ответить на следующие принципиальные вопросы.

Какие данные обрабатывает программа и какова их структура? Обрабатываемые программой данные могут быть достаточно простыми, как переменные различных типов, а могут представлять собой большие массивы взаимосвязанной информации, которые тщательно анализируются и организуются в реляционные базы данных.

Как осуществляется доступ к данным? Данные могут храниться в памяти или на постоянных носителях, и доступ к ним может осуществляться просто по имени или через определенные специально для этого написанные функции. Данные могут быть общедоступными, или же их чтение и изменение может строго регламентироваться.

Каковы принципы именования данных? Применяются ли в данной разработке определенные соглашения об именах? Должны ли имена быть достаточно понятными, чтобы программист, который будет осуществлять сопровождение продукта в дальнейшем, мог по ним понять назначение данных.

Как хранятся данные? Определенные данные будут храниться на постоянном носителе. В каком формате они будут храниться? Как к ним будет осуществляться доступ? Какие для этого понадобятся дополнительные программные средства?

Из всего вышеизложенного о проектировании программной архитектуры неявно следовало, что программный продукт, прежде всего, анализируется с точки зрения его функций, а уже затем — обрабатываемых данных. На самом же деле код и данные представляют собой единое целое. Например, модули, обращающиеся к одним и тем же данным, оказываются связанными самым тесным образом, даже если они выполняют над этими данными совершенно разные операции. Если структура данных

меняется, все обращающиеся к ним модули приходится переписывать. Вот почему полезно строить концепцию программы с позиции обрабатываемых ею данных. С этой точки зрения программа — это нечто, что последовательно преобразует данные от входной информации через ряд промежуточных стадий до выходной информации, которая может, например, представлять собой сгенерированный по запросу пользователя отчет. Модули же рассматриваются как функциональные элементы, необходимые для различных операций, выполняемых над данными: один модуль нужен для ввода информации, другой выполнит над ней конкретные вычисления, а третий выведет результат. Таким образом, модули могут характеризоваться входными и выходными данными и возникать по мере потребности в их преобразовании. При таком анализе выявляются те связи между программными единицами, которые при изучении продукта только с функциональной точки зрения могли бы быть утеряны.

Описание алгоритмов

Проектирование программного продукта не заканчивается описанием программных модулей и организации данных. Необходимо также определить, как именно будут реализовываться поставленные задачи. Обычно за выполнение этой задачи отвечают программисты. Они выбирают оптимальные алгоритмы и описывают (иногда довольно подробно) последовательность логических шагов, необходимых для выполнения каждой задачи.

Моделирование

Это завершающий этап стадии проектирования. Для адекватного представления будущего программного продукта, на этапе проектирования может быть разработан его прототип — программная модель всего продукта или его части. Прототип строится очень быстро и с минимумом затрат. Его очень легко менять, и он не выполняет никакой реальной работы.

Иногда моделирование выполняется не только для внешнего дизайна, но и для внутренней структуры системы. Как было определено выше, при нисходящем проектировании система разбивается на несколько самостоятельных процессов или модулей, которые, в свою очередь, разбиваются на меньшие модули и т. д.

В том же порядке выполняется и их кодирование; первоначально пишутся модули более высокого уровня, а затем создаются модули, которые они вызывают. Однако бывает и так, что подпрограммы нижних уровней в значительной мере определяют всю разработку. Например, системе может потребоваться низкоуровневый обработчик прерывания, вся работа которого выполняется за 60 микросекунд. Этот обработчик разумнее всего будет написать заранее, чтобы убедиться в его целесообразном использовании. В случае неудачи другие модули придется перепроектировать.

Как правило, прототип создается для оценки функциональности будущей системы и ее пользовательского интерфейса. Это исключительно полезная технология: когда пользователи получают возможность собственноручно поэкспериментировать с системой или ее прототипом, их требования могут значительно измениться. При этом идеи, которые в спецификации казались просто блестящими, в работающей модели могут утратить всю свою привлекательность.

Кроме вышеперечисленного в процессе проектирования полезно зафиксировать определение всех используемых в системе понятий с целью их последующего адекватного использования.

1.2.4. Стадия кодирования

После того как определены требования к системе и составлены алгоритмы решений, каждый алгоритм записывается на выбранном языке программирования. В результате программистами создается исходный текст программы. При выборе языковой платформы руководитель разработки должен руководствоваться требованиями, которые предъявляются к системе, и адекватно оценивать возможности программной среды по реализации этих требований. При этом штат компании должен быть укомплектован качественными программистами-кодировщиками, имеющими навыки работы в выбранной среде разработки.

1.2.5. Стадия внедрения

На этой стадии происходит передача заказчику готовой информационной системы. Если система разрабатывается для конкретного заказчика, то внедрение обычно разделяют на два этапа:

1) внедрение системы в опытную эксплуатацию. На этом этапе происходит первое знакомство конечных пользователей с системой. Предполагается, что в ходе опытной эксплуатации пользователями будут выявлены узкие места и недостатки системы. Разработчики, в свою очередь, должны внести необходимые изменения в систему, постоянно поддерживая контакт с пользователями. В завершение первого этапа подписывается акт о сдаче системы в опытную эксплуатацию;

2) внедрение системы в промышленную эксплуатацию. На этом система передается заказчику в полноценное пользование. Для принятия системы в промышленную эксплуатацию создается специальная комиссия, в состав которой входят представители как заказчика, так и исполнителя.

Комиссия после приемочных испытаний системы дает заключение о пригодности системы для промышленной эксплуатации. Также могут быть даны рекомендации по дальнейшему развитию системы. Заключение комиссии фиксируется в специальном протоколе.

В завершение второго этапа сторонами подписывается акт о сдаче системы в промышленную эксплуатацию

1.2.6. Сопровождение системы

После внедрения системы в промышленную эксплуатацию обычно со стороны заказчика возникают пожелания и предложения по дальнейшей модификации системы и дополнению ее различными функциями. Исполнитель в случае заключения договора на сопровождение вносит в систему соответствующие изменения.

Кроме внесения изменений исполнитель на стадии сопровождения консультирует пользователей по вопросам, возникшим в ходе использования системы, и исправляет ошибки, возникшие в системе по своей вине (за свой счет) и ошибки, возникшие в системе по вине заказчика (за дополнительную плату).

Также на этой стадии по договоренности с заказчиком исполнителем проводится обучение пользователей работе с системой.

Каждый этап должен завершаться критическим анализом результатов. Систематическое выполнение критического анализа составляет главную часть процесса проверки соответствия этапов создания информационной системы.

1.3. Определение состава группы по созданию информационных систем

Автоматизированные информационные системы обработки информации и управления промышленного масштаба, отвечающие стандартам качества и представляющие собой законченные надежно функционирующие программные комплексы, редко разрабатываются одиночками: обычно этим занимаются группы людей, иногда довольно многочисленные. В такой группе, называемой ***командой разработчиков,*** у каждого сотрудника своя роль. Даже если приходится создавать программы абсолютно самостоятельно или ограниченным количеством исполнителей, это просто означает, что разработчики по очереди или одновременно выполняют функции всех необходимых членов команды.

Рассмотрим стандартный набор функций, выполняемых членами команды разработчиков, предположив для простоты, что каждая роль принадлежит отдельному сотруднику. На практике в большинстве небольших компаний сотрудники часто выполняют по нескольку функций. Структуру группы (компании) и набор функций рассмотрим на примере крупной фирмы, занимающейся разработкой коммерческих информационных систем (рис. 1.1).

Руководитель компании определяет стратегический план развития компании, цели и задачи организации в целом. Определяет бюджет разработок в целом.

Рис. 1.1. Структура фирмы по разработке ИС

Руководитель проекта (менеджер проекта) отвечает за качество программного продукта, планирование работ и составление и предоставление на утверждение руководителю бюджета разработки. Фактически все отчеты проектировщиков и разработчиков программных продуктов непосредственно передаются ему. В команде планирования программного продукта должны быть ведущие инженеры, персонал, отвечающий за маркетинг и продажи, и руководители проекта. Эта команда вырабатывает общие характеристики продукта. Результатом ее работы является несколько документов, определяющих дальнейшую разработку.

Разработчиков программного продукта может быть несколько и среди них выделяются следующие:

- разработчик архитектуры;
- специалист по анализу предметной области;
- специалист по анализу человеческого фактора;
- программист пользовательского интерфейса;
- ведущие программисты.

Опишем функции специалистов-разработчиков программного продукта на стадии проектирования системы.

Разработчик архитектуры определяет общую внутреннюю структуру кода и данных, принципы обмена данными между связанными программами и их совместного использования, а также стратегию разработки совместно и повторно используемых модулей. Разработчик архитектуры может также составлять план тестирования «стеклянного ящика» на самом верхнем уровне, анализировать технические обзоры всех спецификаций и разрабатывать тесты для приемки продукта, проверяющие соответствие кода техническим требованиям.

Специалист по анализу предметной области должен понимать, чего хотят пользователи и как это выразить в терминах, понятных программисту или другому разработчику.

Специалист по анализу человеческого фактора, или *специалист по эргономике,* имеет психологическое образование и знает, как спроектировать программу, чтобы она была полезной и удобной, и как тестировать продукт (или его прототип) на соответствие этим качествам. Некоторые из специалистов настолько хорошо разбираются в вопросах проектирования и программирования, что могут непосредственно разрабатывать пользовательский интерфейс программ. Остальные специалисты принимают участие в разработке пользовательского интерфейса совместно с программистами.

Программист пользовательского интерфейса специализируется на создании пользовательского интерфейса программ. Обычно это профессиональный программист, который разбирается в оконной архитектуре и компьютерной графике, а в идеальном варианте еще и обладает знаниями в области когнитивной философии.

Пользовательский интерфейс — это часть программы, предоставляющая пользователю информацию в виде графики, текста, звука, в печатном виде и получающая от него ответные данные через клавиатуру, мышь и другие устройства ввода, которые затем передаются для обработки основной программе. Эту часть программы часто называют слоем представления и получения данных. Именно ее и создает программист пользовательского интерфейса.

В более широком смысле понятие пользовательского интерфейса включает также *содержание информации*, которой пользователь обменивается с программой. Например, разработчик пользовательского интерфейса решает, какие опции нужно предоставить пользователю, как понятно их описать и в каком виде отобразить. Хотя многие программисты считают, что могут проектировать пользовательский интерфейс так же хорошо, как и реализовывать, на самом деле большинство из них нуждается в сотрудничестве со специалистом по эргономике.

Ведущие программисты занимаются разработкой той части спецификации (технического задания), которая относится к внутренней структуре продукта. Во многих командах, строящих работу по принципу консенсуса, программисты разрабатывают архитектуру продукта сами.

Менеджер по маркетингу отвечает за соответствие продукта долгосрочной стратегии и имиджу своей компании, а также за маркетинговую деятельность, продолжающуюся после выпуска продукта (рекламу, выпуск и распространение новых версий, обучение продавцов и дилеров). В большинстве компаний менеджер по маркетингу отвечает также и за рентабельность продукта. Он определяет требования рынка, а также те функции и возможности, от которых зависит его конкурентоспособность. В определении набора функций продукта и оборудования, с которым он должен быть совместим, менеджер по маркетингу также играет самую активную роль.

Представитель группы технической поддержки — это член или руководитель группы, непосредственно контактирующей с пользователями. Сотрудники этой группы анализируют жалобы пользователей, отвечают на их вопросы и предоставляют им необходимую информацию. На этапе создания продукта они участвуют в проектировании программы и разработке документации, стараясь сделать ее как можно понятнее и минимизировать количество звонков, на которые им потом придется отвечать.

Технические писатели — это члены ***группы документирования,*** разрабатывающие руководство пользователя и интер-

активную справку. Их советы часто помогают сделать программу более простой и понятной.

Тестировщики также считаются членами команды разработчиков. На самом простом уровне тестировщикам дается задание по проверке функционирования системы, они его выполняют, фиксируя результаты, после чего результаты анализируются. В функции тестировщиков также может входить составление плана тестирования системы.

Кроме перечисленных выше специалистов в разработке информационных систем и других специфических программных проектов принимают участие и другие специалисты: по компьютерной графике, надежности, защите, аппаратному обеспечению, а также юристы, бухгалтера и т. д.

1.4. Проблемы стандартизации нормативов разработки информационных систем

В начальной стадии процесса проектирования АСОИУ перед разработчиком возникает вполне логичный вопрос: во сколько оценить свои трудозатраты. На сегодняшний день в литературе недостаточно рекомендаций по этой части проектирования систем. Одним из стандартов, на основе которого проводятся расчеты по оценке трудоемкости разработки систем, являются нормативы из действующего на настоящий момент отраслевого «Сборника временных норм на работы по ведению Государственного мониторинга геологической среды, информационной деятельности, цифровому картографированию», утвержденного Министерством природных ресурсов РФ 27.12.2000 г.) [1]. Согласно рекомендациям, предложенным в этом сборнике, расчет трудозатрат производится следующим образом.

Объем структуры базы данных информационной системы (ИС) согласно [1] определяется:

- количеством сущностей системы;
- количеством атрибутов, входящих в один объект;
- степенью связанности объектов и атрибутов ИС.

Под сущностью в данном случае будем понимать объект предметной области (одна и более электронных таблиц), эле-

мент информационной системы. Примерами сущностей являются законодательная инициатива, входящий документ, исходящий документ, организация, сотрудники организаций и др.

Под атрибутом сущности (полем электронной таблицы) здесь будем понимать простейший элемент информационной системы, формирующийся вручную или посредством справочников (классификаторов) из элементов первичной однородной фактографической информации, разделенной по назначению.

Жизненный цикл проектирования, создания, внедрения и сопровождения информационных систем описан в ГОСТе 34 «Комплекс стандартов на автоматизированные системы», созданном в конце 1980-х годов как всеобъемлющий комплекс взаимоувязанных межотраслевых документов.

Минимальный набор этапов жизненного цикла системы в целом включает:

- анализ предметной области;
- проектирование ИС в идеологии конкретной СУБД и создание пользовательского интерфейса;
- создание запросов и отчётов для вывода информации в требуемом виде;
- расширение функциональных возможностей ИС;
- тестирование ИС;
- внедрение ИС в эксплуатацию.
- сопровождение системы.

Процедура создания БД описывается выражением, определяющим общее количество полей в наименьшей величине-измерителе.

$$\text{Количество полей} = 2N \times 10K_1 \times 5K_2, \qquad (1.1)$$

где N — коэффициент, отражающий количество объектов ИС;

K_1 — коэффициент, отражающий количество атрибутов на один объект;

K_2 — коэффициент, отражающий количество связей с другими объектами.

Нормализованной величиной при разработке ИС является величина, характеризуемая количеством формируемых полей, входящих в создаваемые таблицы посредством установленных

связей. При значениях коэффициентов, равных единице, величина, выражающая их количество равна 100.

Трудоемкость работ рассчитывается с учетом категорий сложности создания информационных систем. В соответствии с нормативами трудозатраты основных исполнителей по созданию ИС численно равны нормам длительности выполнения этой работы (табл. 1.1)

Таблица 1.1

Нормы длительности создания ИС с применением СУБД

Измеритель — 100 полей

Категория сложности	Значение нормы, смен	Характеристика категории
1 Простая	0,136	Условия создания информационных систем: – использование типовых программных средств и «мастера БД»; – количество прикладных программ индивидуально, но не более трех; – количество полей электронных таблиц до 90000; – использование БД, разработанных в других организациях и заимствованных без изменения; – использование наработок созданных ранее собственных элементов БД
2 Средней сложности	0,194	Условия создания информационных систем: – использование типовых программных средств без применения «мастера БД»; – использование БД, разработанных в других организациях и заимствованных с изменениями, определяемыми организацией разработчиком, но без изменения структуры БД; – количество прикладных программ от трех до десяти; – количество полей электронных таблиц от 90000 до 200000; – использование заимствованных элементов созданных БД

Окончание табл. 1.1

Категория сложности	Значение нормы, смен	Характеристика категории
3 Сложная	0,369	Условия создания информационных систем: – использование языка программирования, совместимого с данной СУБД; – количество прикладных программ не ограничено; – количество полей электронных таблиц от 200000 до 500000; – использование наработок и элементов, не имеющих аналогов и разрабатываемых впервые

Рассмотрим расчет трудоемкости создания ИС на примере. Следует отметить, что расчет трудоемкости в данном случае проводится в целом на создание информационной системы, а не только структурной части базы данных.

Пусть даны исходные данные для расчета трудоемкости разработки программного комплекса, описывающего документооборот организации:

– наличие в создаваемой базе данных не менее 20 составных сущностей;

– среднее число атрибутов (таких как входящий номер документа, входящая дата, тип документа, краткое содержание, резолюция и т. д.) — не менее 35 на одну составную сущность.

Расчет количества полей производится по формуле (1.1):

$$\text{Количество полей} = 2 \times 20 \times 10 \times 35 \times 5 \times 4 = 280000.$$

Таким образом, исходя из коэффициентов, представленных в табл. 1.1, создаваемая система определенно принадлежит к классу сложных систем. Тогда расчет трудоемкости с учетом нормативов табл. 1.1 будет проведен следующим образом:

$$\text{Количество чел.-смен на создание ИС} = 280000 \times 0{,}369 / 100 = 1033{,}2.$$

Проведем также расчет трудозатрат по составлению технической документации. Исходными данными для расчета трудоемкости разработки технической документации являются:

1) состав документации: руководства для пользователей системы, руководство администратора системы, общее описание систем;

2) измеритель — 1 лист формата А4;

3) состав работ — жизненный цикл документирования ИС:

- обобщение материалов;
- структурирование материалов;
- оформление текста сопроводительной документации;
- компьютерный набор текста;

4) количество листов — не менее 90.

Трудозатраты основных исполнителей по составлению документации определяются произведением норм длительности выполнения этой работы на измеритель 0,27 чел.-смен; оператора — 0,12 чел.-см на измеритель:

работы по созданию документации:

$$90 \times 0{,}27 = 24{,}3 \text{ чел.-смен;}$$

работы по компьютерному набору текста:

$$90 \times 0{,}12 = 10{,}8 \text{ чел.-смен.}$$

Таким образом, общие трудозатраты на создание ИС составляют:

$$1033{,}2 + 24{,}3 + 10{,}8 = 1068{,}3 \text{ чел.-смен.}$$

Исходя из полученных данных, мы имеем возможность определить количество исполнителей при известных временных ограничениях, а также размер заработной платы исполнителей.

Так для нашего примера, при необходимости разработать систему за 13 месяцев при 40-часовой рабочей неделе получим примерно 267 рабочих смен. Таким образом, для разработки системы потребуется 4 исполнителя (1068,3/267).

Контрольные вопросы

1. Опишите основные стадии разработки АСОИУ.
2. Перечислите и кратко охарактеризуйте состав группы по созданию информационных систем
3. Поясните механизм определения трудозатрат на создание информационной системы.

2. СТАНДАРТЫ КАЧЕСТВА АСОИУ

2.1. Общая характеристика стандартов качества АСОИУ

Качество — это совокупность характеристик объекта, имеющая отношение к его способности удовлетворить установленные и предполагаемые требования потребителя [3]. Под объектом качества может пониматься как собственно продукция (товары или услуги, информационная технология), процесс ее производства, так и производитель (организация, организационная структура или даже отдельный работник).

Система качества — это совокупность организационной структуры, методик, процессов и ресурсов, необходимых для общего руководства качеством [4].

В соответствии со стандартом ГОСТ Р ISO 9001-96 **система качества** — это структурированный набор документов, регламентирующий определенные аспекты производственной деятельности предприятия. На основании этих документов определяется политика в области качества, осуществляется руководство по качеству, ведется разработка методологических инструкций (описаний процедур) и рабочих инструкций (протоколов мероприятий, форм отчетов, описаний работ и др.).

В целом указанный выше набор документов содержит описание наиболее типичных бизнес-процессов, имеющих отношение к качеству выпускаемой продукции и оказываемых услуг.

Качество информационной системы — обобщенная положительная характеристика системы, выражающая степень ее полезности пользователю.

Для определения функциональной полноты, наличия технических возможностей АСОИУ к взаимодействию, совершенствованию и развитию необходимо использовать стандарты в области оценки показателей их качества [3].

Основой регламентирования показателей качества систем ранее являлся международный стандарт ISO 9126:1991 «Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению». При от-

боре минимума стандартизируемых показателей качества в документе учитывались следующие принципы:

- простота и возможность измерения значений;
- отсутствие перекрытия между используемыми показателями;
- соответствие установившимся понятиям и терминологии;
- возможность последующего уточнения и детализации;
- выделение характеристик, которые позволяют оценивать АСОИУ с позиции пользователя, разработчика и управляющего проектом.

В настоящее время ведутся работы по развитию и совершенствованию этого стандарта в направлении уточнения, детализации и расширения номенклатуры, характеристик качества комплексов программ. Стандарт ISO 9126:1991 заменен на две взаимосвязанные серии стандартов:

ISO 9126:1-4 «Характеристики и метрики качества программного обеспечения»;

ISO 14598-1-6:1998-2000 «Оценивание программного продукта».

Разработанный комплекс стандартов ISO 9126-1-4 (измененная редакция стандарта ISO 9126:1991) состоит из четырех частей под общим заголовком «Информационная технология — Качество программных средств»:

Часть 1: Модель качества;

Часть 2: Внешние метрики;

Часть 3: Внутренние метрики;

Часть 4: Метрики качества в использовании.

Стандарт **ISO 9126-1-4** сохранил прежнюю номенклатуру характеристик качества программных средств. Основные отличия от стандарта ISO 9126:1991 состоят в следующем:

- введены нормативные субхарактеристики на основе информативных субхарактеристик из ISO 9126:1991;
- определены модели качества;
- исключен процесс оценивания, который теперь содержится в ISO 14598-1-6:1998-2000 «Оценивание программного продукта»;
- обеспечена согласованность с ISO 14598-1-6:1998-2000.

В стандарте **ISO 9126-1. Часть 1: Модель качества** рекомендуется специфицировать и оценивать качество систем с различных точек зрения: приобретения, определения требований, разработки, использования, оценивания, поддержки, сопровождения, обеспечения качества, аудита. Модель качества информационной системы, определенная в данной части стандарта, может использоваться для следующих целей:

- проверки полноты определения требований в контракте; идентификации требований к АСОИУ;
- идентификации целей проекта АСОИУ;
- идентификации целей испытаний АСОИУ;
- идентификации критериев приемки пользователем и сертификации законченной разработкой АСОИУ.

Данная часть ISO 9126-1 определяет ***модель характеристик качества***, которая разделяет общее качество информационных систем на шесть базовых характеристик (функциональные возможности, надежность, практичность, эффективность, сопровождаемость и мобильность), далее структурированных на субхарактеристики. Определенные настоящим стандартом характеристики дополнены рядом требований по выбору метрик и их измерению для различных стадий ЖЦ системы.

Метрики (греч. *metrike* — мера, размер) в информационных технологиях — это совокупность принципиально важных показателей, которые определяются и используются для оценки качества программных комплексов. Метрики применимы к любому типу программных систем, включая компьютерные программы и данные, содержащиеся в программируемом оборудовании. Эти характеристики обеспечивают согласованную терминологию для анализа качества информационных систем. Кроме того, они определяют схему для выбора и специфицирования требований к качеству программных систем, а также для сопоставления возможностей различных программных продуктов.

В стандарте **ISO 9126-1. Часть 2: Внешние метрики** используются меры АСОИУ, определенные на основе поведения системы в процессе испытаний, эксплуатации или наблюдения исполняемой системы.

Перед приобретением или использованием системы необходимо провести ее оценку с использованием метрик, учитывающих деловые и профессиональные цели, связанные с использованием, эксплуатацией и управлением продуктом в определенной организационной и технической среде. Внешние метрики обеспечивают заказчикам, пользователям, испытателям и разработчикам возможность определять качество системы в ходе испытаний или эксплуатации.

В требованиях к качеству АСОИУ должны быть перечислены характеристики и субхарактеристики, которые составляют полный набор показателей качества. Процесс формирования требований к качеству завершается определением подходящих внешних метрик, устанавливающих количественные и качественные критерии, которые подтверждают, что разрабатываемая система удовлетворяет потребностям заказчика и пользователя, и их приемлемых диапазонов значений. Далее определяются и специфицируются внутренние атрибуты системы, чтобы спланировать удовлетворение требуемых внешних характеристик качества в конечном продукте и обеспечить их в промежуточных продуктах в ходе разработки.

Подходящие внутренние метрики и их приемлемые диапазоны специфицируются для получения числовых значений или категорий внутренних характеристик качества, чтобы их можно было использовать для проверки соответствия промежуточных продуктов, создаваемых в процессе разработки, внутренним спецификациям качества. Рекомендуется использовать внутренние метрики, которые имеют наиболее сильные связи с целевыми внешними метриками.

Стандарт **ISO 9126-1. Часть 3: Внутренние метрики** применяется в ходе проектирования и программирования к неисполняемым компонентам системы, таким как спецификация или исходный программный текст. При разработке АСОИУ промежуточные продукты оцениваются с использованием внутренних метрик, которые измеряют свойства программ, и могут быть выведены из моделируемого поведения. Основная цель использования внутренних метрик — обеспечить достижение требуемого внешнего качества системы. Внутренние метрики дают

возможность пользователям, испытателям и разработчикам оценивать качество жизненного цикла программ и заниматься вопросами технологического обеспечения качества задолго до того, как АСОИУ становится готовым исполняемым продуктом.

Внутренние метрики позволяют измерять внутренние атрибуты системы или формировать признаки внешних атрибутов путем анализа статических свойств промежуточных или поставляемых программных компонентов. Для измерения внутренних метрик используются категории, числа или характеристики элементов системы, которые, например, имеются в процедурах исходного программного текста, в потоке данных и в представлениях изменения состояний памяти.

Документация также может оцениваться с использованием внутренних метрик.

Стандарт **ISO 9126-1. Часть 4: Метрики качества в использовании** определяет степень удовлетворения продуктом потребностей конкретных пользователей в достижении заданных целей. При этом учитываются: результативность, подразумевающая точность и полноту достижения определенных целей пользователями при применении системы; продуктивность, соответствующую соотношению израсходованных ресурсов и результативности при ее эксплуатации; удовлетворенность — психологическое отношение к качеству используемой системы. Метрики качества в использовании не входят в число шести базовых характеристик АСОИУ (функциональные возможности, надежность, практичность, эффективность, сопровождаемость и мобильность), регламентируемых стандартом ISO 9126-1-4, однако они рекомендуются для интегральной оценки результатов функционирования комплексов программ.

Метрики качества в использовании должны подтверждать качество системы для определенных сценариев и задач. Данные метрики являются оптимальными для определения качества системы пользователем. ***Качество в использовании*** — это восприятие пользователем качества системы, измеряемое скорее в терминах результатов использования системы, чем в показателях собственных внутренних свойств АСОИУ.

Связь качества в использовании с другими характеристиками качества систем зависит от типа пользователя: для конечного пользователя качество в использовании обусловливают в основном характеристики функциональных возможностей, надежности, практичности и эффективности, а для персонала сопровождения АСОИУ качество в использовании определяется сопровождаемостью.

На качество в использовании могут влиять любые характеристики качества, и это понятие шире, чем, например, практичность, которая связана с простотой использования и привлекательностью. Качество в использовании в той или иной степени характеризуется сложностью применения комплекса программ, которую можно описать трудоемкостью использования с требуемой результативностью. Многие характеристики и субхарактеристики показателей качества АСОИУ обобщенно отражаются неявными технико-экономическими показателями, которые определяют функциональную пригодность конкретной АСОИУ.

Из всего вышеизложенного нетрудно заметить, что в течение последних десятилетий активизировался рост числа всевозможных стандартов качества информационно-программных систем. Исходя из этого, следует учитывать, что постоянно меняющиеся стандарты не всегда позволяют адекватно оценить реальное качество разрабатываемых систем.

2.2. Стандартизированные показатели качества информационных систем

Стандарты категоризуют атрибуты качества системы по шести характеристикам:

1) функциональным возможностям;
2) надежности;
3) практичности;
4) эффективности;
5) сопровождаемости;
6) мобильности.

Далее характеристики подразделяются на субхарактеристики, которые могут измеряться внутренними или внешними метриками.

Исходя из принципиальных возможностей их измерения, все характеристики объединены в три группы (табл. 2.1):

1) ***категорийно-описательные***, отражающие набор свойств и общие характеристики объекта (функции, категории ответственности, защищенности и важности), которые могут быть представлены номинальной шкалой категорий;

2) ***количественные***, представляемые множеством упорядоченных, равноотстоящих точек, отражающих непрерывные закономерности, и описываемые интервальной или относительной шкалой. Эти показатели можно объективно измерить и численно сопоставить с требованиями;

3) ***качественные***, содержащие несколько упорядоченных или отдельных значений (категорий), которые характеризуются порядковой или точечной шкалой набора категорий, устанавливаются в значительной степени субъективно и экспертно.

Функциональные возможности — способность АСОИУ обеспечивать функции, удовлетворяющие установленным потребностям заказчиков и пользователей при применении комплекса программ в заданных условиях. Данная характеристика определяет, какие функции и задачи решает АСОИУ для удовлетворения потребностей, в то время как другие характеристики главным образом связаны непосредственно с функционированием системы. В данной характеристике для установленных и подразумеваемых потребностей применимы описания и примечания к определению характеристик качества. Субхарактеристики функциональной возможности можно охарактеризовать в основном категориями и качественным описанием функций, для которых трудно определить меры и шкалы. Поэтому они отнесены в отдельную группу категорийно-описательных метрик.

Функциональная пригодность — это набор и описания атрибутов, определяющих назначение, номенклатуру, основные, необходимые и достаточные функции АСОИУ, заданные техническим заданием и спецификациями требований заказчика или потенциального пользователя. В процессе проектирования АСОИУ атрибуты функциональной пригодности должны конкретизироваться в спецификациях на компоненты и на систему в целом.

Некоторые атрибуты можно численно представить точностью результатов, относительным числом поэтапно изменяемых функций, числом реализуемых требований спецификаций заказчиков и т. д. Кроме них функциональную пригодность отражают множество различных специализированных метрик, которые тесно связаны с конкретными функциями и областью применения программ.

Таблица 2.1

Характеристики качества АСОИУ

Категорийно-описательные метрики	
Функциональные возможности	Функциональная пригодность Корректность (правильность) Способность к взаимодействию Защищенность Согласованность
Количественные метрики	
Надежность	Завершенность Устойчивость к дефектам Восстанавливаемость Доступность (готовность)
Эффективность	Временная эффективность Используемость ресурсов
Качественные метрики	
Практичность	Понятность Простота использования Изучаемость Привлекательность
Сопровождаемость	Анализируемость Изменяемость Стабильность Тестируемость
Мобильность	Адаптируемость Простота установки Сосуществование (соответствие) Замещаемость

В наибольшей степени функциональная пригодность связана с *корректностью и надежностью* АСОИУ. Кроме них

функциональная пригодность отражается множеством различных характеристик и субхарактеристик, таких как способность компонентов к взаимодействию и степень стандартизации интерфейсов, мобильность программ и их защищенность от негативных внешних воздействий.

Корректность (правильность) — способность системы обеспечивать правильные или приемлемые результаты и эффекты. Данное понятие включает получение ожидаемых данных с необходимой степенью точности расчетных значений. Приведенные ниже виды корректности используются в основном для интегральной оценки разработанных информационных систем.

В процессе проектирования модулей и групп программ применяются частные конструктивные показатели корректности, которые включают корректность структуры программ, обработки данных и межмодульных интерфейсов. Каждый из частных показателей может характеризоваться несколькими методами измерения качества и достигаемой степенью корректности программ: детерминировано, стохастически или в реальном времени. Корректность программных модулей включает функциональную и конструктивную корректность.

Конструктивная корректность модулей заключается в соответствии их структуры общим правилам структурного программирования и конкретным правилам оформления и внутреннего строения программных модулей в данном проекте. Функциональная корректность модулей определяется корректностью обработки исходных данных и получения результатов.

Корректность обработки данных также имеет функциональную и конструктивную составляющие. Конструктивная корректность обработки данных определяется правилами их структурирования и упорядочения. Эти правила могут быть достаточно полно формализованы без учета конкретных особенностей функций программ.

Назначение и область применения программ определяют выбор используемых структур данных и конкретных методик их упорядочения. Функциональная корректность обработки данных связана в основном с конкретизацией их содержания в процессе

исполнения программ, а также учитывается при подготовке данных внешним абонентам.

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

Способность к взаимодействию — свойство АСОИУ и их компонентов взаимодействовать с одной или большим числом указанных систем или компонентов. Способность программных и информационных компонентов к взаимодействию можно оценивать объемом изменений в системе, которые необходимо выполнить при дополнении или исключении некоторой функции, при отсутствии изменений операционной или аппаратной среды.

Этот показатель связан с такими субхарактеристиками, как корректность и унифицированность межмодульных интерфейсов, которая определяется двумя видами связей: по управлению и по информации.

Связи по управлению определяются вызовами программных модулей и возвратами в вызывавшие модули. Взаимодействие модулей по информации может происходить через обменные переменные, непосредственно подготавливаемые и используемые соседними модулями, или через глобальные переменные между более крупными компонентами. Многообразие и сложность информационных связей в крупных системах значительно затрудняют формализацию и измерение достигнутой корректности взаимодействия программ.

Защищенность — способность систем защищать программы, информацию и данные. В критериях защиты и обеспечения безопасности для конкретной АСОИУ сосредоточиваются разнообразные характеристики, которые в ряде случаев трудно или невозможно описать количественно, в связи с чем приходится оценивать их экспертно или по бальной системе.

Основное внимание в практике обеспечения безопасности применения информационных систем сосредоточено на защите от злоумышленных разрушений, искажений и хищений про-

граммных средств и информации баз данных. При этом подразумевается наличие заинтересованных лиц в доступе к конфиденциальной или полезной информации с целью ее незаконного использования, хищения, искажения или уничтожения.

В реальных сложных системах возможны катастрофические последствия и аномалии функционирования, отражающиеся на безопасности применения, при которых их источниками являются случайные, непредсказуемые, дестабилизирующие факторы при отсутствии непосредственно заинтересованных в подобных нарушениях лиц.

Наиболее полно качество защиты ИС характеризуется величиной предотвращенного ущерба, возможного при проявлении дестабилизирующих факторов и реализации конкретных угроз безопасности, а также средним временем между возможными проявлениями угроз, нарушающих безопасность. Однако описать и измерить возможный ущерб при нарушении безопасности для критических АСОИУ разных классов практически невозможно. Поэтому факты реализации угроз целесообразно характеризовать интервалами времени между их проявлениями или наработкой на отказы, отражающиеся на безопасности. Это сближает понятия и характеристики степени безопасности с показателями надежности АСОИУ. Принципиальное различие состоит в том, что в показателях надежности учитываются все реализации отказов, а в характеристиках защиты следует регистрировать только те отказы, которые отразились на безопасности функционирования.

Достаточно универсальным измеряемым показателем при этом остается длительность восстановления нормальной работоспособности информационной системы. Приближенно такие катастрофические отказы в восстанавливаемых системах можно выделять по превышению некоторой допустимой длительности восстановления работоспособности.

В требованиях к программам, обеспечивающим защиту, следует отражать все аспекты, необходимые для удовлетворения согласованных потребностей заказчика по общей безопасности АСОИУ.

Согласованность — соответствие системы стандартам, нормативным документам, соглашениям или нормам законов и другим предписаниям, связанным с функциями, областью применения и защитой АСОИУ.

Надежность — свойство комплекса программ обеспечивать достаточно низкую вероятность отказа в процессе функционирования системы в реальном времени. Основные термины и определения, имеющие отношение к надежности АСОИУ, изложены в третьем разделе. Здесь же отметим, что надежность функционирования программ является понятием динамическим, проявляющимся во времени.

Завершенность — свойство системы не попадать в состояния отказов вследствие имеющихся ошибок и дефектов в программах и данных. Количество необнаруженных дефектов и ошибок непосредственно отражается на длительности нормального функционирования комплекса программ между сбоями и отказами.

Завершенность можно характеризовать наработкой (длительностью) на отказ при отсутствии автоматического восстановления (рестарта), измеряемой обычно часами. На этот показатель надежности влияют только отказы вследствие проявившихся дефектов.

Устойчивость к дефектам и ошибкам — свойство системы поддерживать заданный уровень качества функционирования в случаях проявления дефектов и ошибок или нарушений установленного интерфейса. Для этого в систему должна вводиться временная, программная и информационная избыточность, реализующая оперативное обнаружение дефектов функционирования, их идентификацию и автоматическое восстановление нормального функционирования системы.

Относительная доля вычислительных ресурсов, используемых непосредственно для быстрой ликвидации последствий отказов и оперативного восстановления нормального функционирования системы (рестарта) отражается на повышении устойчивости и надежности программ. Наработка на отказ при наличии оперативного рестарта определяет величину устойчивости.

Восстанавливаемость — свойство системы в случае отказа восстанавливать заданный уровень качества функционирования, а также поврежденные программы и данные. После отказа системы иногда бывает неработоспособны в течение некоторого периода времени, продолжительность которого определяется его восстанавливаемостью. Основным показателем процесса восстановления является длительность восстановления и ее вероятностные характеристики.

Восстанавливаемость характеризуется также полнотой восстановления нормального функционирования программ в процессе ручного или автоматического их перезапуска. Перезапуск должен обеспечивать возобновление нормального функционирования системы, на что требуются ресурсы ЭВМ и время. Поэтому полнота и длительность восстановления функционирования после сбоев и отказов определяет надежность системы и возможность его использования по прямому назначению.

***Доступность* (*готовность*)** — свойство системы выполнять требуемую функцию в данный момент времени при заданных условиях использования. Доступность может оцениваться временем, в течение которого система находится в работоспособном состоянии, в пропорции к общему времени применения. Следовательно, доступность связана с такими субхарактеристиками, как завершенность, устойчивость к ошибкам и восстанавливаемость, которые в совокупности обусловливают длительность простоя после каждого отказа, а также длительность наработки на отказ. Для определения этой величины измеряется время работоспособного состояния системы между последовательными отказами или началами нормального функционирования системы после них.

Готовность системы характеризуется ***коэффициентом готовности,*** который отражает вероятность иметь восстанавливаемую систему в работоспособном состоянии в произвольный момент времени.

Эффективность — свойство системы обеспечивать требуемую производительность с учетом количества используемых вычислительных ресурсов в установленных условиях. Эти ресурсы могут включать другие программные продукты, аппаратные

средства, средства телекоммуникации и т. п. Таким образом, эффективность характеризуется долей времени использования средств вычислительной техники для решения основных функциональных задач системы.

Временная эффективность — свойство системы обеспечивать требуемое время отклика и обработки заданий, а также пропускную способность при выполнении его функций в заданных условиях. Временная экономичность системы определяется длительностью выполнения заданных функций. Она зависит от скорости обработки данных, влияющей непосредственно на интервал времени завершения конкретного вычислительного процесса, и от пропускной способности, т. е. от числа заданий, которое можно реализовать на данной ЭВМ в заданном интервале времени.

Данные показатели качества тесно связаны со временем реакции (отклика) системы на запросы при решении основных функциональных задач. Величина этого времени зависит от длительности решения задачи центральным процессором ЭВМ, затрат времени на обмен с внешней памятью, на ввод и вывод данных и длительности ожидания в очереди до начала решения задачи.

Временная эффективность обусловливается с длительностью обработки типового запроса или интервалом времени решения типовых или наиболее частых функциональных задач данными системами.

Пропускная способность комплекса программ на конкретной ЭВМ отражается числом сообщений или запросов на решение определенных задач, обрабатываемых в единицу времени. Она зависит от функционального содержания системы и конструктивной его реализации и может рассматриваться как один из внутренних показателей качества программ. Декомпозицию этого показателя целесообразно проводить, ориентируясь на конкретные функции системы и особенности ее архитектуры.

Используемость ресурсов — свойство системы использовать доступные вычислительные ресурсы в течение заданного времени при выполнении его функций в установленных условиях. Эта субхарактеристика может быть определена показателем ре-

сурсной экономичности, который отражает количество и степень занятости ресурсов центрального процессора, оперативной, внешней и виртуальной памяти, каналов ввода-вывода, терминалов и каналов локальной сети. Этот показатель определяется структурой и функциями системы, а также архитектурными особенностями и доступными ресурсами ЭВМ. В зависимости от конкретных особенностей системы и ЭВМ при выборе критериев может доминировать либо величина абсолютной занятости ресурсов различных видов, либо относительная величина использования ресурсов каждого вида при нормальном функционировании системы.

Ресурсная экономичность влияет не только на стоимость решения функциональных задач, но, зачастую, особенно для встраиваемых ЭВМ, определяет принципиальную возможность полноценного функционирования системы в условиях реально ограниченных вычислительных ресурсов.

Несмотря на быстрый рост доступных ресурсов памяти и производительности ЭВМ, очень часто потребности в ресурсах для решения конкретных задач системой превышают имеющееся их количество, что актуализирует задачу оценки и экономного использования вычислительных ресурсов.

Такие характеристики, как практичность, сопровождаемость и мобильность в основном определяются качественными оценками. Для некоторых субхарактеристик сопровождаемости и мобильности могут доминировать технико-экономические меры трудоемкости (человеко-часы) и длительности (часы), используемые для характеристики процесса решения конкретных задач. Однако для многих показателей в этой группе характеристик приходится применять порядковые меры экспертных бальных шкал с небольшим числом (2–4) градаций.

Практичность (применимость) — свойство системы, характеризующееся сложностью ее понимания, изучения и использования, а также привлекательность для пользователя при применении в указанных условиях. В число пользователей могут быть включены операторы, конечные пользователи и косвенные пользователи, непосредственно не связанные с системой.

В практичности следует учитывать всё разнообразие характеристик внешней среды пользователей, на которые может влиять система, включая возможную подготовку к использованию и оценку результатов функционирования программ.

Практичность (применимость) использования системы – понятие достаточно абстрактное и трудно формализуемое, однако в итоге зачастую значительно определяющее функциональную пригодность и полезность применения системы. В эту группу показателей входят критерии, с различных сторон отражающие функциональную понятность, удобство освоения, эффективность или простоту использования. Некоторые из субхарактеристик можно оценивать затратами труда и времени на их реализацию.

Понятность — свойство системы, обеспечивающее пользователю возможность определения степени пригодности ее для конкретных задач и имеющихся условий эксплуатации. Даная характеристика определяется качеством документации и первичными впечатлениями от системы в целом.

Понятность системы может быть охарактеризована четкостью функциональной концепции, широтой демонстрационных возможностей, полнотой и наглядностью представления в документации возможных функций, распознаваемостью модифицируемых параметров и адаптируемостью системы к конкретной среде и условиям применения пользователями.

Простота использования определяется возможностью и комфортностью эксплуатации и управления системой. Свойства изменяемости, адаптируемости и легкости установки могут быть предпосылками для простоты использования. Простота использования характеризуется управляемостью, устойчивостью к ошибкам и согласованностью с ожиданиями пользователя.

Эта субхарактеристика учитывает физические и психологические характеристики пользователей и отражает уровень комфортности условий эксплуатации системы, которым присущи простота управления функциями системы и высокая информативность сообщений пользователю, наглядность и унифицированность управления экраном, а также доступность изменения функций в соответствии с квалификацией пользователя и чис-

лом операций, необходимых для запуска определенного задания и анализа результатов.

Кроме того, простота использования характеризуется рядом динамических параметров: временем ввода и отклика на задание, длительностью решения типовых задач, временем на регистрацию результатов.

Изучаемость характеризуется удобством изучения системы пользователем с целью ее применения. Она определяется трудоемкостью и длительностью подготовки пользователя к полноценной эксплуатации системы. Эти показатели зависят от возможности предварительного обучения и совершенствования в процессе эксплуатации, от возможностей оперативной помощи и подсказки при использовании системы, а также от полноты, доступности и удобства использования руководств и инструкций по эксплуатации.

Изучаемость может также характеризовать объем (число страниц) эксплуатационной документации и/или объем (Кбайт) электронных учебников.

Привлекательность — субъективное свойство системы «нравиться» пользователям. Оно связано с внешними атрибутами оформления системы и эксплуатационной документации, обусловливающими большую или меньшую его привлекательность для пользователя.

Сопровождаемость – приспособленность системы к модификации и изменению конфигурации. Изменения могут включать исправления, усовершенствования или адаптацию системы к изменениям в среде применения, а также в требованиях и функциональных спецификациях заказчика.

Простота и трудоемкость модификаций определяются внутренними характеристиками качества комплекса программ, которые отражаются на внешнем качестве и качестве в использовании, а также на сложности управления конфигурацией системы.

Анализируемость — способность системы к диагностике ее дефектов или причин отказов, а также к идентификации и выделению ее компонентов для модификации. Эта субхарактеристика зависит от стройности архитектуры, унифицированно-

сти интерфейсов, полноты и корректности технологической и эксплуатационной документации на систему.

Изменяемость – приспособленность системы к достаточно простой реализации специфицированных изменений и к управлению конфигурацией. Реализация модификаций включает кодирование, проектирование и документирование изменений, которые характеризуются определенной трудоемкостью и временем, связанным с исправлением дефектов и/или модернизацией функций, а также с изменением условий эксплуатации.

В оценках этой субхарактеристики учитываются влияние структуры, интерфейсов и технических особенностей системы и не рассматриваются воздействия крупных, принципиальных изменений ее функций.

Если систему должен модифицировать конечный пользователь, изменяемость может быть предпосылкой и частью простоты использования.

Стабильность — способность системы предотвращать и минимизировать непредвиденные негативные эффекты от ее изменений. Эта внутренняя субхарактеристика определяется архитектурой системы, унифицированностью интерфейсов, корректностью технологической документации и может существенно влиять на функциональную пригодность, надежность и адекватность поведения системы при использовании и усовершенствовании.

Тестируемость – способность системы обеспечивать простоту проверки изменений и приемки модифицированных компонентов программ. Оценки этой субхарактеристики зависят от четкости правил структурного построения компонентов и всего комплекса программ, унификации межмодульных и внешних интерфейсов, полноты и корректности технологической документации.

Возможность локализации изменений и унификация интерфейсов компонентов с некорректируемой частью системы позволяет сокращать сложность, трудоемкость и длительность их тестирования, упрощает подготовку тестов и анализ результатов.

В этой субхарактеристике отражаются в основном технические составляющие процесса тестирования модификаций без

учета организационной и функциональной части их подготовки. Обобщенно ее можно оценивать затратами труда и времени на тестирование некоторых средних модификаций программ.

Мобильность — приспособленность системы к переносу из одной аппаратно-операционной среды в другую. Переносимость программ и данных на различные аппаратные и операционные платформы является важным свойством функциональной пригодности для многих современных систем, которое может оцениваться объемом, трудоемкостью и длительностью необходимых доработок компонентов системы и операций по адаптации, которые следует выполнить для обеспечения полноценного функционирования системы после переноса на иную платформу.

Мобильность должна быть заложена на уровне исходных текстов программ или на уровне объектного кода. Она зависит от структурированности и расширяемости комплексов программ и данных, а также от дополнительных ресурсов, необходимых для реализации переносимости и модификации компонентов при их переносе.

Адаптируемость — способность системы к модификации для эксплуатации в различных аппаратных и операционных средах без применения других дополнительных действий или средств. Адаптируемость включает масштабируемость внутренних возможностей (например, экранных полей, размеров таблиц, объемов транзакций, форматов отчетов и т. д.). В случае адаптации системы конечным пользователем эта субхарактеристика может быть компонентом для определения простоты использования.

Простота установки — способность системы к простому внедрению (инсталляции) в указанной среде заказчика или пользователя. В случае установки системы конечным пользователем конечным пользователем легкость (простота) установки может быть компонентом для определения удобства использования. Также как и адаптируемость данная характеристика может измеряться трудоемкостью и длительностью процедур установки, а также степенью удовлетворения требованиям заказчика и пользователей.

Сосуществование **(*соответствие*)** — способность системы сосуществовать и взаимодействовать с другими независи-

мыми системами в общей вычислительной среде, разделяя общие ресурсы. Эта субхарактеристика зависит от степени стандартизации интерфейсов системы с операционной и аппаратной средой применения и может оцениваться экспертно.

Замещаемость — приспособленность системы к относительно простому использованию других различных компонентов вместо выделенных, подлежащих замене. Замещаемость не предполагает, что заменяемый компонент системы способен полностью изменить сущность рассматриваемой системы. Она может включать атрибуты, как простоты установки, так и адаптируемости. Большую роль для этого свойства играют четкая структурированность архитектуры и стандартизация внутренних и внешних интерфейсов системы. Замещаемость характеризуется трудоемкостью и длительностью замены крупных компонентов системы.

2.3. Показатели качества баз данных

Неотъемлемой составляющей большинства информационных систем является база данных (БД), в которой хранится вся доступная пользователю системы информация. Для баз данных пока отсутствуют международные стандарты, регламентирующие показатели качества. Ниже представлен набор характеристик, который наиболее часто используется на практике при выборе и оценке баз данных.

В системах баз данных доминирующее значение приобретают сами данные, их хранение и обработка. Поэтому при анализе качества БД целесообразно выделить два компонента:

1) программные средства системы управления базой данных (СУБД), независимые от сферы их применения и смыслового содержания накапливаемых и обрабатываемых данных;

2) информацию базы данных, доступную для обработки и использования в конкретной проблемно-ориентированной сфере применения.

Практически весь набор показателей качества АСОИУ, изложенный выше, в той или иной степени может использоваться при анализе и оценке качества программ СУБД. Особенности

состоят в изменении акцентов при выборе и упорядочении этих показателей качества.

Почти во всех случаях важнейшими показателями качества СУБД являются функциональные характеристики процессов формирования и изменения информационного наполнения БД администраторами, а также доступа к данным и представления результатов пользователям БД.

Качество интерфейса специалистов с БД, обеспечиваемого средствами СУБД, оценивается в значительной степени субъективно, однако имеется ряд характеристик, которые можно оценивать достаточно корректно.

Различия требований к показателям качества привели к созданию весьма широкого спектра локальных, специализированных и распределенных СУБД. Специализированные СУБД характеризуются относительно узкой сферой применения и более четким выделением доминирующей группы показателей качества. В универсальных СУБД спектр показателей качества шире, что позволяет соответственно расширять сферу применения конкретного типа СУБД.

Вторым компонентом БД является собственно информация, накапливаемая и обрабатываемая в базе данных. Выделяемые показатели качества должны представлять практический интерес для пользователей БД и быть упорядоченными в соответствии с приоритетами их применения. Кроме того, каждый выделяемый для проверки показатель должен быть пригоден для достаточно достоверного измерения и сравнения с требуемым значением при испытаниях и сертификации.

Функциональными показателями качества информации БД являются:

- полнота накопленных описаний объектов (относительное число объектов или документов, имеющихся в БД, к общему числу объектов по данной тематике или к числу объектов в аналогичных БД по той же тематике);
- достоверность (степень соответствия данных об объектах в БД реальным объектам вне ЭВМ в данный момент времени, определяющаяся изменениями самих объектов, некорректностями записей об их состоянии или некорректностями расчетов

их характеристик);

- идентичность данных (относительное число описаний объектов, не содержащих ошибок, к общему числу документов об объектах в БД;
- актуальность данных (относительное число устаревших данных об объектах в БД к общему числу накопленных и обрабатываемых данных).

К **конструктивным показателям качества информации в БД** относятся в основном объемно-временные характеристики сохраняемых и обрабатываемых данных:

- объем базы данных — число записей описаний объектов или документов в базе данных, доступных для хранения и обработки;
- оперативность — степень соответствия динамики изменения данных в процессе сбора и обработки состояниям реальных объектов или величина запаздывания между появлением или изменением характеристик реального объекта и его отражением в базе данных;
- периодичность — промежуток времени между поставками двух последовательных, достаточно различающихся информацией версий БД;
- глубина ретроспективы — интервал времени от даты выпуска и/или записи в базу данных самого раннего документа до настоящего времени;
- динамичность — относительное число изменяемых описаний объектов к общему числу записей в БД за некоторый интервал времени, определяемый периодичностью издания версий БД.

Кроме того, к конструктивным относятся все показатели защищенности информации. Защищенность реализуется в основном программными средствами СУБД, однако в сочетании с поддерживающими их средствами организации данных.

В распределенных базах данных показатели защищенности тесно связаны с характеристиками целостности данных. Эти показатели отражают степень тождественности данных в памяти удаленных компонентов распределенной БД.

Защита информации в ИС является серьезной и кропотливой задачей. На самом элементарном уровне ее решение сводит-

ся к обеспечению выполнения двух фундаментальных мероприятий: проверка полномочий пользователя (*санкционирование доступа*) и проверка подлинности (*аунтификация*) [5].

Проверка полномочий пользователя предполагает определение для каждого пользователя набора санкционированных действий, которые он может выполнять по отношению к определенным объектам БД. В большинстве СУБД задание и проверка полномочий определяется внутренними средствами системы, одним из способов задания полномочий является использование операторов Grant и Revoke языка SQL.

Выделяются следующие уровни доступа пользователей к объектам БД:

- создание объекта БД;
- изменение;
- чтение;
- удаление;
- администрирование (полный доступ к объекту).

Сведения о полномочиях пользователя находятся в защищенной области базы данных.

Проверка подлинности заключается в достоверном подтверждении того, что пользователь, выполняющий санкционированные действия, действительно является тем, за кого себя выдает [5]. В любой СУБД должна поддерживаться модель проверки подлинности, которая может обеспечить надежную верификацию идентификаторов, предъявляемых пользователями. Обычно контроль подлинности пользователя, работающего с базой данных, заключается в сопоставлении вводимых пользователем имени и пароля при входе в систему с эталонными идентификаторами пользователя, сохраненными в БД.

Одним из способов безопасного хранения данных является использование *модели многоуровневой безопасности данных* [5], такая система безопасности означает, что в системе хранится информация, относящаяся к разным классам безопасности, при этом пользователь может иметь доступ только к уровню, определенному конкретно для него, и нижним уровням. Многоуровневая безопасность в отношении БД может строиться на основе *модели Белла-ЛаПадула* [5], при этом объекты базы данных под-

вергаются классификации (например, особо секретно, секретно, конфиденциально, для общего пользования), а каждый пользователь причисляется к одному из уровней допуска к классам объектов. Механизмы обеспечения безопасности данных постоянно совершенствуются разработчиками СУБД и являются на сегодняшний день одним из перспективных направлений в развитии информационных технологий.

При реальном функционировании БД важную роль играют временные характеристики взаимодействия конечных пользователей и администраторов БД в процессе эксплуатации базы данных по прямому назначению. Эти характеристики зависят от качества СУБД, а также от объема, структуры и других рассмотренных выше показателей качества используемой информации.

Для баз данных важнейшим ресурсом является оперативная и внешняя память ЭВМ, занимаемая информацией БД, а также активность использования этих ресурсов при работе пользователя в системе. Эти показатели качества влияют на время реакции БД на разные виды запросов пользователей и на пропускную способность БД при эксплуатации.

2.4. Выбор характеристик и метрик качества АСОИУ

Для выбора показателей качества и сравнения с требованиями спецификаций, а также для сопоставления их значений между различными программными продуктами необходимы определенные метрики. Стандартами рекомендуется, чтобы было предусмотрено измерение каждой характеристики качества АСОИУ (или субхарактеристики) с точностью и определенностью, достаточной для установления критериев и выполнения сравнений, и чтобы эта точность обеспечивалась при измерении.

Следует предусматривать нормы допустимых ошибок измерения, вызванных инструментами измерений и ошибками человека. Метрики, используемые для сравнений, должны быть утверждены, и иметь точность, достаточную для выполнения надежных сравнений. Для этого требуется, чтобы измерения были объективны, воспроизводимы, и чтобы эмпирические измерения использовали интервальную или еще лучшую шкалу.

Чтобы измерения были объективны, должна быть документирована и согласована процедура для присвоения числового значения или категории каждому атрибуту программного продукта. При эмпирических измерениях должны выполняться наблюдения или психометрически одобренные вопросники с применением номинальной, интервальной или порядковой шкалы. Процедуры измерений должны давать в результате одинаковые меры (с приемлемой устойчивостью), получаемые различными субъектами при выполнении одних и тех же измерений.

Для внутренних метрик целесообразно учитывать связь каждой из них с некоторым требуемым внешним критерием. Внутренняя мера конкретного атрибута АСОИУ должна находиться в определенном соотношении с некоторым измеримым аспектом качества системы. Важно, чтобы измерения соответствовали значениям, совпадающим с нормальными, очевидными предположениями (например, если измерение показывает, что продукт имеет высокое качество, то этому должна соответствовать характеристика продукта, полностью удовлетворяющая конкретные потребности пользователя).

Выше отмечалось, что показатели качества систем с позиции возможности и точности их измерения можно разделить на три типа, особенности которых полезно уточнить для обеспечения их выбора.

К категорийно-описательным относятся показатели качества, которые характеризуются наибольшим разнообразием значений свойств программ и наборов данных и охватывают весь спектр классов, назначений и функций современных АСОИУ. Эти свойства можно сравнивать только в пределах однотипных систем и трудно упорядочивать по принципу предпочтительности. Среди стандартизированных показателей качества к этому типу, прежде всего, относится функциональная пригодность, являющаяся самой важной и доминирующей характеристикой любых АСОИУ.

Номенклатура и значения всех остальных показателей качества непосредственно определяются требуемыми функциями программного средства и в той или иной степени влияют на выполнение этих функций. Поэтому выбор функциональных воз-

можностей системы, их подробное и максимально корректное описание являются исходными данными для установления всех остальных стандартизированных показателей качества.

К ***количественным*** стандартизированным показателям качества относятся достаточно достоверно и объективно измеряемые характеристики: надежность и эффективность. Значения этих характеристик обычно в наибольшей степени влияют на функциональные возможности и метрики в использовании системы. Поэтому выбор и обоснование требуемых их значений должны проводиться наиболее аккуратно и достоверно уже при системном проектировании АСОИУ. Их субхарактеристики могут быть описаны упорядоченными шкалами объективно измеряемых значений, требуемые численные величины которых могут быть установлены и выбраны заказчиками или пользователями системы.

Надежность может отражаться наработкой на отказ, измеряемой чаще всего часами, средним временем восстановления, обычно в пределах секунд или минут, а также коэффициентом готовности — вероятностью застать систему в работоспособном состоянии при нормальной эксплуатации.

Эти величины могут выбираться и фиксироваться в техническом задании или спецификации требований и сопровождаться методикой объективных, численных измерений при квалификационных испытаниях для сопоставления с требованиями. Для каждой из них может быть установлен допустимый разброс численных значений и требуемая точность измерений.

Показатели временной эффективности тесно связаны между собой и влияют на функциональную пригодность системы. Длительность решения основных задач, пропускная способность по числу их решений за некоторый интервал времени, длительность ожидания результатов (отклика) и некоторые другие характеристики динамики функционирования системы могут быть выбраны и установлены количественно в спецификациях требований заказчиком. Эта субхарактеристика не всегда может быть выбрана и достаточно точно зафиксирована на начальных этапах разработки, но она может количественно измеряться и последовательно уточняться в жизненном цикле системы.

Используемость ресурсов ЭВМ, если она не достигает критических значений, когда некоторого ресурса становится недостаточно, менее существенно влияет на функциональную пригод-ность системы. Однако избыток ресурсов снижает экономическую эффективность ИС и должен сохраняться в минимальной степени. Выбор и количественное измерение степени использования различных ресурсов ЭВМ может значительно влиять на изменение функциональной пригодности системы.

Группа ***качественных*** стандартизированных показателей включает практичность, сопровождаемостъ и мобильность. Эти характеристики АСОИУ трудно полностью описать измеряемыми количественными значениями и их субхарактеристики в основном имеют описательный, качественный вид.

В зависимости от функционального назначения АСОИУ экспертно по согласованию с заказчиком можно определять степень необходимости этих свойств и бальные значения их атрибутов в жизненном цикле конкретной системы. Например, не всегда может предусматриваться требование мобильности программ при их переносе на иные операционные и аппаратные платформы и производиться выбор и оценка соответствующих субхарактеристик.

Сопровождаемость может быть определена как неполная замена системы на вновь разработанные версии и тем самым сливаться с процессами разработки или осуществляться как непрерывная поддержка множества пользователей консультациями, а также адаптациями и корректировками программ. При этом может быть определена трудоемкость процессов сопровождения, которая используется как обобщенная качественная характеристика при выборе и установлении требований к этому показателю качества.

В зависимости от функции различаются и трудоемкость процессов сопровождения, которая может использоваться как обобщенная качественная характеристика при выборе и установлении требований к этому показателю качества. Соответственно качественно могут быть установлены субхарактеристики сопровождаемости и описаны требуемые их свойства.

Практичность наиболее тесно связана с функциональной пригодностью. Обобщенно этот показатель можно отразить трудоемкостью и длительностью, которые необходимы для изучения и полного освоения функций и технологии применения соответствующей системы.

Каждая из субхарактеристик практичности имеет ряд качественных атрибутов, которые отмечены выше. Эти показатели могут выбираться и оцениваться экспертно с учетом функционального назначения информационной системы, а также надежности и ресурсной эффективности комплекса программ.

Процессы выбора и установления метрик и шкал для описания показателей качества системы можно разделить на два этапа:

1) выбор и обоснование набора исходных данных, характеризующих общие особенности и область применения системы, а также этапы жизненного цикла проектирования системы, каждый из которых влияет на определенные характеристики качества комплекса программ;

2) выбор, установление и утверждение конкретных метрик и шкал измерения показателей качества проекта для их последующего оценивания и сопоставления с требованиями в процессе квалификационных испытаний или сертификации на определенных этапах жизненного цикла АСОИУ.

На ***первом этапе*** в качестве основы следует использовать всю базовую номенклатуру характеристик и субхарактеристик, стандартизированных в ISO 9126-1-4. Их описания желательно предварительно упорядочить по приоритетам с учетом сферы применения проекта системы. Далее необходимо выделить и ранжировать по приоритетам потребителей, которым необходимы определенные показатели качества конкретного проекта системы с учетом их специализации и профессиональных интересов.

Широкая номенклатура характеристик, представленная в стандарте ISO 9126-1-4, поддерживает разнообразные требования, из которых следует выбирать необходимые с позиции потребителей этих данных. Среди потребителей, для которых не-

обходим выбор и установление показателей качества программных средств выделеляются:

1) пользователи или подразделения предприятий-пользователей, предпочитающие оценивать качество и пригодность системы, используя реализуемый набор функций и обобщенные метрики качества, важные при использовании;

2) заказчики, требующие производить оценивание системы, прежде всего, по значениям показателей, определяющих общую сферу применения системы (функциональные возможности, надежность, практичность и эффективность), и заданных в спецификации требований;

3) коллектив специалистов, сопровождающий систему и устанавливающий приоритеты оценок на основе метрик сопровождаемости, обычно независимо от функционального назначения;

4) лица, ответственные за реализацию системы в различных аппаратных и операционных средах, оценивающие систему с использованием метрик мобильности;

5) разработчики, технологи-инструментальщики и специалисты системы качества, поддерживающие жизненный цикл системы, отдающие приоритет значениям внутренних метрик каждой характеристики качества.

Первые две группы потребителей заинтересованы в установлении внешних показателей качества в процессе использования конечного, готового программного продукта. Для этих потребителей при выборе важно выделить и по возможности формализовать внешние, эксплуатационные характеристики и метрики программного средства на завершающих этапах жизненного цикла системы. Эти данные должны быть формализованы в контракте, техническом задании и спецификации требований заказчика, согласованных с разработчиками. Внутренние метрики качества для них являются второстепенными.

Для остальных трех групп потребителей важными являются характеристики системы на промежуточных этапах ЖЦ, на которых проявляются в основном внутренние, технологические свойства комплекса программ. Эти характеристики качества АСОИУ могут интересовать заказчика и требовать с его сто-

роны формализации постольку, поскольку они обеспечивают качество конечного продукта при применении. Они должны формализоваться для осуществления контроля в соответствии с принятой на данном предприятии системой качества, а также для технологического обеспечения качества в течение всего ЖЦ системы. Их можно не представлять в составе эксплуатационной документации для пользователей и отражать только в технологической документации разработчиков, специалистов по сопровождению и переносу программ и данных, а также предоставлять заказчикам по специальному запросу.

Приоритеты потребителей при выборе показателей качества отражаются не только на выделении важнейших для них критериев и ранжировании характеристик, но также на исключении из анализа некоторых показателей качества, которые для данного потребителя не имеют значения.

Подготовка исходных данных завершается выделением номенклатуры базовых, приоритетных показателей качества, определяющих функциональную пригодность системы для определенных потребителей.

Среди важнейших показателей качества, которые необходимо установить и формализовать в исходных данных, чаще всего являются функциональные возможности для соответствующей сферы применения системы. Эта характеристика и ее субхарактеристики, с учетом особенностей потребителей, доминируют в последующем выборе показателей для определения качества АСОИУ для конкретного потребителя.

На ***втором этапе*** после фиксирования исходных данных, которое должен выполнить потребитель, процессы выбора номенклатуры и метрик начинаются с ранжирования характеристик и субхарактеристик для конкретного проекта. Этот анализ должны проводить специалисты, обеспечивающие ЖЦ комплекса программ и реализацию установленных показателей качества совместно с заказчиком и пользователями.

Далее этими специалистами для каждого из отобранных показателей должна быть согласована и установлена метрика и шкала оценок субхарактеристик и их атрибутов для проекта и потребителя результатов анализа. Там, где возможны количе-

ственные измерения или оценки качества, кроме номинальных, требуемых значений может потребоваться выбор и установление допусков на отклонения от величин, требуемых спецификациями.

Для показателей, представляемых качественными признаками, желательно определить и зафиксировать в спецификациях описания условий, при которых следует считать, что данная характеристика реализуется в системе.

Выбранные значения характеристик качества и их атрибутов должны быть предварительно проверены разработчиками на их реализуемость с учетом доступных ресурсов конкретного проекта и при необходимости откорректированы.

Результаты анализа и выбора номенклатуры и метрик характеристик качества проекта системы должны быть документированы в спецификациях требований, согласованы с их потребителями и утверждены заказчиком проекта.

2.5. Сравнение качества АСОИУ по критерию функциональной полноты

Необходимость проведения сравнительного анализа программных продуктов возникает как перед потенциальным пользователем в случае приобретения системы, так и перед разработчиком при создании собственной системы с целью изучения уже существующих наработок в этой предметной области. Для такого анализа необходимы или рабочие копии конкурирующих продуктов, или, по крайней мере, их демонстрационные версии или описания, если ничего больше достать не удастся. Составляется перечень их функций, сильных и слабых сторон и тех характеристик, которые отмечаются в прессе и профессиональных изданиях как достоинства и недостатки этих продуктов. Производится классификация продуктов.

Продукты разделяются по занимаемым ими сегментам рынка или по специфическим назначениям. Затем составляется детальный отчет обо всех продуктах, включая и те, появление которых на рынке только предполагается. В отчет включается четко структурированное описание каждого продукта, и такое же описание составляется для будущего продукта компании.

На основании отобранных таким образом данных можно ответить на ключевой вопрос проводимого анализа — какая из систем является предпочтительной в использовании.

Ниже приводится методика выбора (оценки) автоматизированных информационных систем, основанная на проверке соответствия функциональной полноты системы требованиям пользователя или некоторому эталону [6].

Пусть $Z = \{Z_i\}$ ($i = 1, 2, \ldots, n$) — множество сравниваемых АИС;

$R = \{R_j\}$ ($j = 1, 2, \ldots, m$) — множество, составляющее словарь реализуемых АИС функций $\{Z_i\}$.

Исходная информация представляется в виде таблицы $\{X_{ij}\}$, элементы которой определяются следующим образом:

$$X_{ij} = \begin{cases} 1, \text{если } j-\text{я функция реализуется } i-\text{й АИС;} \\ 0, \text{если не реализуется.} \end{cases}$$

Выделим системы Z_i и Z_k ($i, k = 1, 2, \ldots, n$) и введем следующие обозначения:

$P_{ik}^{(11)}$ — число функций, выполняемых и Z_i и Z_k, то есть

$P_{ik}^{(11)} = |Z_i \cap Z_k|$ — мощность пересечения множеств $Z_i = \{X_{ij}\}$ и $Z_k = \{X_{kj}\}$ ($j \in m; x | x_{ij} \wedge x_{kj} = 1$);

$P_{ik}^{(10)}$ — число функций, выполняемых Z_i, но не реализуемых Z_k, то есть

$P_{ik}^{(10)} = |Z_i \backslash Z_k|$ — мощность разности множеств $Z_i = \{X_{ij}\}$ и $Z_k = \{X_{kj}\}$;

$P_{ik}^{(01)}$ — число функций, выполняемых Z_k но не реализуемых Z_i, то есть

$P_{ik}^{(01)} = |Z_k \backslash Z_i|$ — мощность разности множеств Z_k и Z_i;

$P_{ik}^{(00)} = |Z_i \cup Z_k|$ — мощность объединения множеств Z_i и Z_k, то есть

$$P_{ik}^{(00)} = P_{ik}^{(11)} + P_{ik}^{(10)} + P_{ik}^{(01)}.$$

Для оценки того, какая часть (доля) функций, выполняемых АИС Z_i, реализуется также АИС Z_k можно использовать следующую величину:

$$H_{ik} = P_{ik}^{(11)} / (P_{ik}^{(11)} + P_{ik}^{(10)}), \quad (0 \le H_{ik} \le 1).$$

Взаимосвязь между АИС Z_i и Z_k оценивается по значениям $P_{ik}^{(11)}$ и $G_{ik} = P_{ik}^{(11)} / P_{ik}^{(00)}$, $(0 \le G_{ik} \le 1)$, где G_{ik} — «мера подобия».

Выбирая различные пороговые значения матриц G и H, можно построить логические матрицы поглощения (включения) G_0, H_0. Например, элементы матрицы H_0 получим следующим образом:

$$H_{ik}^0 = \begin{cases} 1, \text{если } H_{ik}^0 \ge \varepsilon_h, i \ne k; \\ 0, \text{если } H_{ik}^0 < \varepsilon_h, \text{или } i = k. \end{cases}$$

$$G_{ik}^0 = \begin{cases} 1, \text{если } G_{ik}^0 \ge \varepsilon_g, i \ne k; \\ 0, \text{если } G_{ik}^0 < \varepsilon_g \text{ или } i = k. \end{cases}$$

Граф, построенный по логическим матрицам G^0 и H^0, дает наглядное представление о взаимосвязи между сравниваемыми АИС (по выполняемым функциям).

Строку с перечнем функций, которые в идеале должна выполнять система, обозначим через Z_e.

Дополнив таблицу $\{X_{ij}\}$ ($i \in n$, $j \in m$) строкой X_{ej} ($j \in m$), рассчитаем матрицы $P^{(01)}$, $P^{(11)}$ и, выделив строки, у которых $P_{ej}^{(10)} = 0$, получим перечень АИС, полностью удовлетворяющих требованиям к функциональной полноте программного средства.

Приведем пример экспериментального исследования методики. Для этого определим функции и параметры информационных систем автоматизированного документооборота, наиболее широко представленных на рынке.

Характеристики описанных ниже систем определялись на основе материалов открытой печати, изданий по компьютерной тематике (Мир ПК, Открытые Системы, Computerworld Россия, PC Week/RE, КомпьютерПресс и др.) материалов конференций,

выставок, семи-наров; рекламных материалов фирм-производителей; материалов размещаемых в сети Интернет, а также на основе сведений представленных в [7–11].

Система автоматизации делопроизводства и документо-оборота *«Дело» ЗАО «Электронные офисные системы»* предназначена для автоматизации делопроизводственной деятельности, основанной на традиционных отечественных технологиях и закрепленной соответствующими стандартами, и документационного обеспечения управленческой деятельности государственных организаций. Система поддерживает полный жизненный цикл документа в организации от создания проекта документа до списания в дело и передачи в архив. При работе с проектом документа выполняется последовательная или параллельная маршрутиза-ция, контролируются сроки рассмотрения и срок подготовки проекта в целом.

Система автоматизации делопроизводства и документооборота *«LanDocs» фирмы АО «Ланит»* выполнена в соотвествии с отечественными стандартами и нормами, с использованием практики организации учета документов и контроля исполнения поручений в организациях раз-личных типов и предоставляющая разные уровни функцио-нальности для различных категорий сотрудников.

Система «LanDocs» реализована как адаптивная CASE-модель электронного офисного документооборота и делопроизводства. Настройка системы на конкретные условия эксплуатации осуществляется модификацией параметров CASE-моделей без изменения программного кода. Наличие подсистемы автоматизированной поддержки деловых процессов организа-ции обеспечивает возможность проектирования и управления исполнением предопределенных маршрутных (Workflow) схем обработки документов

Программно-технологический комплекс *«Золушка» НТЦ «Институт развития Москвы»* представляет собой технологию классического делопроизводства со сквозным контролем исполнения документов. Эта технология позволяет автоматизировать основные функции канцелярии, общего или организа-

ционного отдела — регистрацию, обработку и контроль исполнения документов.

Система обеспечивает автоматизацию традиционных функций делопроизводства — от регистрации исходящей и входящей почты в многоуровневой карточке до организации ведения архива электронных копий документов.

Электронная канцелярия состоит из 3-х функциональных компонент программных систем: «Служебная корреспонденция», «Решения и распоряжения», «Письма граждан». Комплекс позволяет организовать работы на разнородных программных и аппаратных платформах (от DOS и Windows до LotusNotes).

Существуют реализации системы для всех современных клиент-серверных платформ (Oracle, MS SQL Server, Lotus Notes/Domino и др.). Файл-серверная версия не требует покупки и установки дополнительного программного обеспечения и работает в промышленном режиме при объеме до 20000 зарегистрированных документов в год.

Файл-серверная версия системы эффективна при внедрении информационных технологий в небольших организациях. При этом возможен переход от файл-серверной к любой из клиент-серверных версий системы. Этот переход является преемственным, поскольку происходит с сохранением всех данных и пользовательского интерфейса.

Реализации системы на платформах Oracle, в Lotus Domino/Notes, в MS SQL предназначены для внедрения в территориально-распределенных организациях с многоуровневой системой управления.

Система автоматизированного документооборота *«КОРД» НИИ автоматики и электромеханики* (*г. Томск*) предназначена для автоматизации документооборота, делопроизводственной деятельности и процесса управления в организациях различных форм собственности. В системе обеспечивается автоматизированный контроль соблюдения регламента исполнения документов. Реализована функция анализа исполнительской дисциплины сотрудников организации.

Программный комплекс (ПК) «КОРД» состоит из семи функциональных компонент — автоматизированных рабочих

мест. Каждый АРМ программного комплекса рассчитан на эксплуатацию конкретным подразделением организации.

Благодаря модульности системы, а также возможности работы в однопользовательской конфигурации, обеспечена возможность внедрения ПК «КОРД» в организациях с небольшим штатом сотрудников либо в организациях со слабой компьютерной инфраструктурой. Возможность сосредоточения всех функций системы в одном АРМе позволяет возложить на работника канцелярии учет делопроизводства и документооборота, а также обеспечение контроля исполнительской дисциплины.

Использование в системе блока поддержки принятия решений на основе накопленных статистических сведений позволяет сделать руководителю и помощнику по контролю вывод о целесообразности передачи документа на исполнение конкретному сотруднику, вовремя передать документ на исполнение другому сотруднику, добавить соисполнителя и определить новые контрольные сроки.

Программное обеспечение ПК «КОРД» разработано в двух вариантах: в архитектуре файл-сервер под управлением MS Access (97/XP); в архитектуре клиент-сервер, где в качестве СУБД используется Oracle 9i. Следует отметить наличие однотипного интерфейса системы в файл-серверной и клиент-серверной конфигурации.

В табл. 2.2. перечислены параметры и функции описанных выше систем, а также параметры и функции так называемой системы эталона, наличие которых в системе делопроизводства и документооборота способствует полной автоматизации этих процессов в организации.

Таблица 2.2

Сводная таблица параметров и функций систем автоматизации документооборота и делопроизводства

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон
	Виды документов, регистрируемых в системе					
1.	Входящие	1	1	1	1	1
2.	Исходящие	1	1	1	1	1
3.	Внутренние	1	1	1	1	1
4.	Обращения граждан	1	1	0	1	1
	Общие реквизиты регистрационной карточки					
5.	Регистрационный номер документа	1	1	1	1	1
6.	Дата регистрации	1	1	1	1	1
7.	Код рубрики темы	1	1	0	1	1
8.	Краткое содержа-ние документа	1	1	1	1	1
9.	Номер дела	1	1	1	1	1
10.	Ключевые слова	0	0	0	1	0
11.	Реквизиты резолю-ции по документу	1	1	1	1	1
12.	Реквизиты конт-рольной службы	1	1	1	1	1
13.	Реквизиты архив-ного хранения	1	1	1	0	1
	Реквизиты организации-корреспондента					
14.	Наименование органи-зации-корреспондента	1	1	0	1	1
15.	Исходящий номер	1	1	1	1	1
16.	Исходящая дата	1	1	1	1	1
17.	Подпись	1	1	1	1	1
	Регистрация входящих документы					
18.	Кому адресован	1	1	0	1	1
19.	Вид доставки	1	1	1	0	1
20.	Отметка о наличии приложений (связан-ные документы)	1	1	1	1	1
21.	Признак повторности	1	1	1	1	1
22.	Тип документа	1	0	0	0	1

Продолжение табл. 2.2

№	Параметры	Системы автоматизации делопроизводства и документооборота

		КОРД	Дело	LanDocs	Золушка	Система эталон
	Регистрация сопроводительные документы					
23.	Аннотация	1	1	1	1	1
24.	Корреспондент	1	1	1	1	1
25.	Исходящий номер	1	1	1	1	1
26.	Исходящая дата	1	1	1	1	1
27.	Кто подписал	1	1	0	1	1
28.	Исполнитель	1	0	0	1	1
	Регистрация писем и обращений граждан					
29.	Корреспондент	1	1	0	0	1
30.	Признак коллективности	1	1	0	0	1
	Регистрация исходящих документов					
31.	Кому адресован	1	1	0	1	1
32.	Кто подписал	1	1	0	1	1
33.	Подразделение- автор	1	1	0	1	1
34.	ФИО исполнителя	1	1	0	1	1
35.	Ссылка на номер входящего документа	1	1	1	1	1
36.	Ссылка на документ	1	0	0	1	1
37.	Вид отправки	1	0	1	0	1
	Контроль исполнения документов					
38.	Сведения о исполнителе	1	1	1	1	1
39.	Гриф утверждения	1	1	1	0	1
40.	Текст задания	1	1	1	0	1
41.	Контролер	1	0	1	1	1
	Выделение ответственного исполнителя	1	1	1	0	1
42.	Методы предупреж-дающего контроля и механизм поддержки принятия решений	1	0	0	0	1
	Сроки исполнения документов					
43.	Поступление к исполнению	1	1	1	1	1
44.	Плановый срок	1	1	1	1	1
45.	Фактический срок	1	1	1	1	1
46.	Напоминание для просроченных	1	1	0	1	1

Окончание табл. 2.2

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон
	Поиск документов					
	Поиск по атрибутам регистрационной карточки					
47.	Группа документов	1	1	1	1	1
48.	Дата документа	1	1	0	1	1
49.	Тематический рубрикатор	1	1	0	1	1
50.	Фильтры поиска	1	1	0	1	1
51.	Критерии поиска для входящих	1	1	0	1	1
52.	Критерии поиска для исходящих	1	1	0	1	1
	Поиск по регистрационным номерам					
53.	Группа документов	1	1	1	1	0
54.	Номер документа	1	1	1	1	1
	Год регистрации	1	1	0	1	1
55.	Подразделение	0	0	0	1	0
	Формирование отчетов					
56.	Сведения о документообороте за заданный период времени	1	1	0	1	1
57.	Сводка об исполнении контрольных документов	1	1	1	1	1
58.	Справка-напоминание об исполнении контрольных документов	1	1	0	1	1

По вышеописанному алгоритму рассчитаем следующие матрицы:

$$P^{(01)} = \begin{vmatrix} 0 & 0 & 0 & 2 & 0 \\ 6 & 0 & 2 & 5 & 5 \\ 24 & 20 & 0 & 22 & 24 \\ 10 & 7 & 6 & 0 & 10 \\ 2 & 1 & 2 & 3 & 0 \end{vmatrix}; \quad P^{(10)} = \begin{vmatrix} 0 & 6 & 24 & 10 & 2 \\ 0 & 0 & 20 & 7 & 1 \\ 0 & 2 & 0 & 6 & 2 \\ 2 & 5 & 22 & 0 & 3 \\ 0 & 5 & 24 & 10 & 0 \end{vmatrix};$$

$$P^{(11)} = \begin{vmatrix} 56 & 52 & 32 & 41 & 56 \\ 52 & 52 & 32 & 45 & 51 \\ 32 & 32 & 34 & 28 & 32 \\ 41 & 45 & 28 & 50 & 47 \\ 56 & 51 & 32 & 47 & 56 \end{vmatrix}; \quad P^{(00)} = \begin{vmatrix} 56 & 58 & 56 & 53 & 58 \\ 58 & 52 & 54 & 57 & 57 \\ 56 & 54 & 34 & 56 & 58 \\ 53 & 57 & 56 & 50 & 60 \\ 58 & 57 & 58 & 60 & 56 \end{vmatrix}.$$

При использовании порогового значения ε_h = 0,8 получим логическую матрицу поглощения H^0.

$$H = \begin{vmatrix} 1 & 0{,}9 & 0{,}7 & 0{,}84 & 0{,}97 \\ 1 & 1 & 0{,}6 & 0{,}87 & 0{,}98 \\ 1 & 0{,}94 & 1 & 0{,}82 & 0{,}94 \\ 0{,}96 & 0{,}9 & 0{,}56 & 1 & 0{,}92 \\ 1 & 0{,}91 & 0{,}57 & 0{,}82 & 1 \end{vmatrix}; \quad H^0 = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

При использовании порогового значения ε_g = 0,75 получим логическую матрицу подобия G^0.

$$G = \begin{vmatrix} 1 & 0{,}9 & 0{,}58 & 0{,}77 & 0{,}97 \\ 0{,}9 & 1 & 0{,}6 & 0{,}8 & 0{,}9 \\ 0{,}58 & 0{,}6 & 1 & 0{,}5 & 0{,}55 \\ 0{,}77 & 0{,}8 & 0{,}5 & 1 & 0{,}77 \\ 0{,}97 & 0{,}9 & 0{,}55 & 0{,}77 & 1 \end{vmatrix}; \quad G^0 = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

По матрицам G^0 и H^0 построим графы подобия (рис. 2.1) и поглощения (рис. 2.2), соответственно.

Из полученных графов можно сделать вывод, что при выбранных коэффициентах подобия и поглощения системами, в наибольшей мере отвечающими требованиям к технологии документооборота и делопроизводства, являются системы «КОРД» и «Дело». Однако при этом необходимо отметить, что в данном случае были выбраны средние коэффициенты подобия и поглощения (ε_g = 0,75 и ε_h = 0,8).

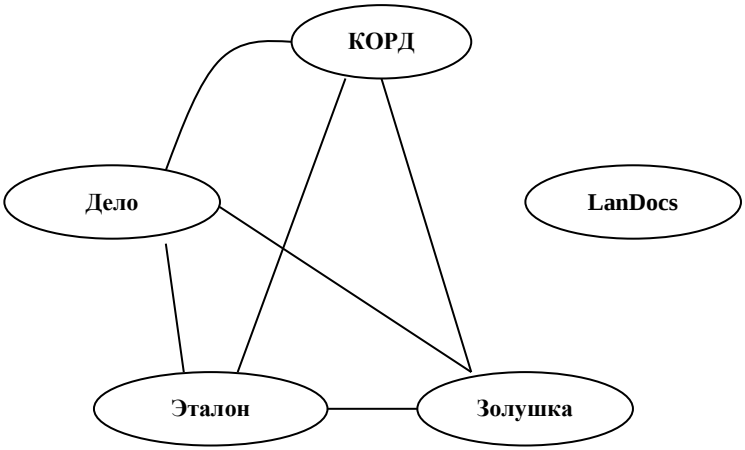

Рис. 2.1. Граф подобия

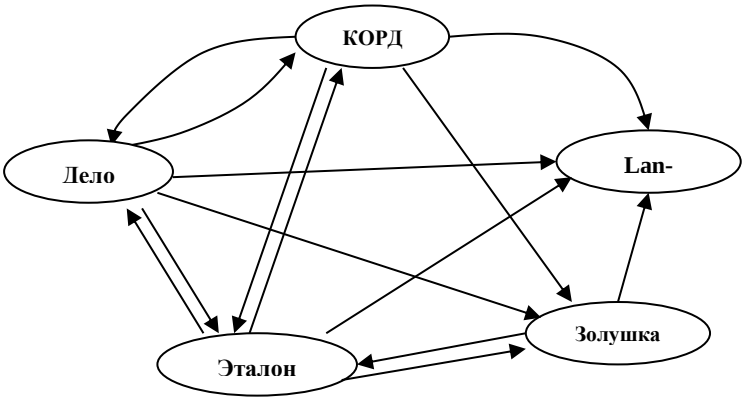

Рис. 2.2. Граф поглощения

В принципе допускается варьирование коэффициентов $0{,}5 \leq \varepsilon_g \leq 1$ и $0{,}5 \leq \varepsilon_h \leq 1$. Явно видно, что при использовании максимальных коэффициентов условия подобия и поглощения не соблюдаются, а при наименьших значениях предложенных коэффициентов все рассматриваемые системы в целом могут быть использованы для автоматизации документооборота и делопроизводства. С целью выбора системы, в наибольшей мере отве-

чающей требованиям потребителя, необходимо выбирать коэффициенты подобия и поглощения близкими к единице.

Применение рассмотренной выше методики позволяет проводить сравнительный анализ любых однотипных автоматизированных информационных систем и делать вывод о предпочтении использования системы и ее соответствии требованиям пользователя или системе эталону.

Контрольные вопросы

1. Дайте определение понятий качества и системы качества.
2. Опишите основные положения стандартов серии ISO: 9000.
3. Перечислите и кратко опишите стандартизированные показатели качества информационных систем и баз данных.
4. Поясните процесс проведения сравнения АСОИУ по критерию функциональной полноты.

3. НАДЕЖНОСТЬ АСОИУ

3.1. Основные положения теории надежности

Надежность является основным показателем качества информационных систем обработки информации и управления. В данном разделе подробно рассматриваются общие положения теории надежности и методы повышения надежности АСОИУ. Согласно ГОСТ 13377-55 **надежность** определяется, как свойство объекта выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонта, хранения и транспортирования.

Основные свойства надежности изучаются теорией надежности, описывающей математические модели, методы и алгоритмы, направленные на изучение проблем предсказания, оценки и оптимизации различных показателей надежности. В более общем смысле, «теория надежности устанавливает закономерности возникновения отказов устройств и методы их прогнозирования; изыскивает способы повышения надежности изделий при конструировании и последующем изготовлении, а также приемы поддержания надежности во время их хранения и эксплуатации; разрабатывает методы проверки надежности при приемке больших партий продукции» [4]. Теория надежности вводит в рассмотрение ***количественных показателей качества систем***.

В ГОСТ 15467-79, ГОСТ 13377-55, ГОСТ 27.002-89 приведены основные термины теории надежности:

безотказность — свойство объекта непрерывно сохранять работоспособное состояние в течение некоторого времени или периода наработки;

долговечность — свойство объекта сохранять работоспособное состояние при установленной системе технического обслуживания и ремонта;

ремонтопригодность — свойство объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания и ремонта;

сохраняемость — свойство объекта сохранять в заданных пределах значения параметров, характеризующих способность объекта выполнять требуемые функции, в течение периода хранения и (или) транспортирования.

Вышеперечисленные свойства надежности характеризуют определенные технические состояния объекта. Различают пять основных видов технического состояния объектов:

1) ***исправное состояние,*** при котором объект соответствует всем требованиям нормативно-технической и (или) конструкторской (проектной) документации;

2) ***неисправное состояние,*** при котором объект не соответствует хотя бы одному из требований нормативно-технической и (или) конструкторской (проектной) документации;

3**) *работоспособное состояние,*** при котором значения всех параметров, характеризующих способность объекта выполнять заданные функции, соответствуют требованиям нормативно-технической и (или) конструкторской (проектной) документации;

4) ***неработоспособное состояние***, при котором значения хотя бы одного параметра, характеризующего способность объекта выполнять заданные функции, не соответствует требованиям нормативно-технической и (или) конструкторской (проектной) документации;

5) ***предельное состояние,*** при котором дальнейшая эксплуатация объекта недопустима или нецелесообразна, либо восстановление его работоспособного состояния невозможно или нецелесообразно.

Переход объекта (изделия) из одного вышестоящего технического состояния в нижестоящее обычно происходит вследствие событий: повреждений или отказов. Совокупность фактических состояний исследуемого объекта и возникающих событий, способствующих переходу в новое состояние, охватывает так называемый **жизненный цикл (ЖЦ)** объекта, который протекает во времени и имеет определенные закономерности, изучаемые в теории надежности.

Согласно ГОСТ 27.002-89 **отказ** — это событие, заключающееся в нарушении работоспособного состояния объекта; **повреждение** — событие, заключающееся в нарушении исправно-

го состояния объекта при сохранении работоспособного состояния.

В ГОСТ 15467-79 введено еще одно понятие, отражающее состояние объекта, — дефект. **Дефектом** называется каждое отдельное несоответствие объекта установленным нормам или требованиям. *Дефект отражает состояние отличное от отказа*. В соответствии с определением отказа как события, заключающегося в нарушении работоспособности, предполагается, что до появления отказа объект был работоспособен. Отказ может быть следствием развития неустраненных повреждений или наличия дефектов: царапин; потертости изоляции; небольших деформаций.

Оценка надежности представляет собой измерение количественных метрик атрибутов субхарактеристик в использовании: завершенности, устойчивости к дефектам, восстанавливаемости и доступности/готовности (подробнее см. раздел 2).

Надежность технических систем определяется двумя главенствующими факторами: надежностью компонент и дефектами в конструкции, допущенными при проектировании или изготовлении [3]. Эти же факторы учитываются и при оценке надежности АСОИУ, при этом определяющими являются ошибки, возникающие на стадии проектирования систем, поскольку надежность систем при их функционировании (физическая целостность) зависит от человеческого фактора и надежности носителя информации (жесткого магнитного диска, CD и т. д.). При этом надежность системы при обработке данных зависит от того, насколько безошибочно спроектирована и технически качественно реализована система. Следует учитывать и возможность самой системы влиять на бесперебойную работу вычислительной техники.

3.2. Основные показатели надежности АСОИУ

При использовании понятия надежности в области разработки АСОИУ следует учитывать особенности и отличия объектов таких систем от традиционных технических систем [3]:

- не для всех видов ИС применимы понятия и методы теории надежности. Данные понятия и методы можно использовать

только для АИС, функционирующих в реальном времени и непосредственно взаимодействующих с внешней средой;

• для повышения надежности комплексов программ особое значение имеют методы автоматического сокращения длительности восстановления и преобразования отказов в кратковременные сбои путем введения в программные компоненты элементов временной, программной и информационной избыточности;

• относительно редкое разрушение программных компонент и необходимость их физической замены приводит к принципиальному изменению понятий сбоя и отказа программ и к разделению их по длительности восстановления относительно некоторого допустимого времени простоя для функционирования АИС;

• непредсказуемость места, времени и вероятности проявления дефектов и ошибок, а также их редкое обнаружение при реальной эксплуатации достаточно надежных программных средств не позволяет эффективно использовать традиционные методы априорного расчета показателей надежности сложных систем, ориентированных на стабильные измеряемые значения надежности составляющих компонент;

• традиционные методы форсированных испытаний надежности систем путем физического воздействия на их компоненты не применимы для АСОИУ и их следует заменять методами форсированного воздействия информационных потоков внешней среды на систему.

Таким образом, с учетом вышеперечисленного можно сформулировать задачи теории и анализа надежности АСОИУ [3]:

• формирование основных понятий, используемых при исследовании и применении показателей надежности программных средств;

• выявление и исследование основных факторов, определяющих характеристики надежности сложных программных комплексов;

• выбор и обоснование критериев надежности для комплексов программ различного типа и назначения;

• исследование дефектов и ошибок, динамики их изменения при отладке и сопровождении;

• исследование и разработка методов структурного построения АСОИУ, обеспечивающих необходимую надежность;

• исследование методов и средств контроля и защиты от искажений программ, вычислительного процесса и данных путем использования различных видов избыточности и помехозащиты;

• разработка методов и средств определения и прогнозирования характеристик надежности в жизненном цикле АСОИУ с учетом их функционального назначения, сложности, структурного построения и технологии разработки.

Решение этих задач позволяет обеспечить создание АСОИУ с заданными показателями надежности. Основными характеристиками надежности АСОИУ являются: отсутствие ошибок, устойчивость к ошибкам, возможность перезапуска системы.

Надежность любой программной системы тем выше, чем реже в ней происходят сбои, особенно такие, которые приводят к потере информации.

Надежная программа должна обеспечивать достаточно низкую вероятность отказа в процессе функционирования в реальном времени [3]. Быстрое реагирование системы на искажения программ, данных или вычислительного процесса и восстановление работоспособности за время меньшее, чем порог между сбоем и отказом, характерные для корректно выполненных программ, обеспечивают ее высокую надежность. При этом и некорректная программа может функционировать абсолютно надежно. В реальных условиях по разным причинам исходные данные могут попадать в области значений, вызывающих сбои, не проверенные при испытаниях, а также не заданные требованиями спецификации и технического задания. Если в этих ситуациях происходит достаточно быстрое восстановление и не фиксируется отказ, то такие события не влияют на основные показатели надежности — наработку на отказ и коэффициент готовности. Следовательно, надежность функционирования программ является понятием динамическим, проявляющимся во времени.

Непредсказуемость вида, места и времени проявления дефектов системы в процессе эксплуатации приводит к необходимости создания специальных, дополнительных систем опера-

тивной защиты от непредумышленных, случайных искажений вычислительного процесса, программ и данных.

Системы оперативной защиты предназначены для выявления и блокирования распространения негативных последствий проявления дефектов и уменьшения их влияния на надежность функционирования АСОИУ до устранения их первичных источников. Для этого в системе должна вводиться временная, программная и информационная избыточность, осуществляющая оперативное обнаружение дефектов функционирования, их идентификацию и автоматическое восстановление (рестарт) нормального функционирования АСОИУ. *Надежность системы должна повышаться за счет средств обеспечения помехоустойчивости, оперативного контроля и восстановления функционирования программ и баз данных.* Эффективность такой защиты зависит от используемых методов, координированности их применения и выделяемых вычислительных ресурсов на их реализацию [3].

Надежность функционирования АСОИУ характеризуется устойчивостью, или способностью к безотказному функционированию, и восстанавливаемостью работоспособного состояния после произошедших сбоев или отказов. **Устойчивость системы** зависит от уровня критичности неустраненных дефектов и ошибок и способности АСОИУ реагировать на их проявления так, чтобы это не отражалось на показателях надежности, что определяется эффективностью контроля данных, поступающих из внешней среды, и средств обнаружения аномалий функционирования АСОИУ [3].

Восстанавливаемость характеризуется полнотой и длительностью восстановления функционирования программ в процессе перезапуска системы (рестарта). Перезапуск должен обеспечивать возобновление нормального функционирования системы, на что требуются ресурсы ЭВМ и время. Поэтому полнота и длительность восстановления функционирования после сбоев отражают качество и надежность АСОИУ и возможность его использования по прямому назначению [3].

Для определения надежности АСОИУ используется **критерий длительности наработки на отказ**, который определяется

временем «работоспособного состояния системы между последовательными отказами или началами нормального функционирования системы после них» [3]. Вероятностные характеристики этой величины также используются в качестве критериев надежности, учитывая возможность многократных отказов и восстановлений. Для оценки надежности восстанавливаемых систем важную роль играют «характеристики функционирования после отказа в процессе восстановления» [3].

Основным показателем процесса восстановления является длительность восстановления и ее вероятностные характеристики. Этот показатель учитывает возможность многократных отказов и восстановлений. Обобщение характеристик отказов и восстановлений выражается в показателе коэффициента готовности, который отражает вероятность того, что система будет находиться в работоспособном состоянии в произвольный момент времени. Значение коэффициента готовности соответствует доле времени полезной работы системы на достаточно большом интервале, содержащем отказы и восстановления.

Интенсивность отказов λ — один из наиболее важных в теоретическом и практическом отношении показателей, который определяет законы распределения вероятностей отказов. В результате многочисленных исследований и обработки статистических испытаний установлено, что распределение вероятностей отказов подчиняется экспоненциальному закону надежности, параметром которого является λ. Таким образом, интенсивность отказов характеризует частоту появления отказов. В практических целях пользуются статистической оценкой появления отказов [12], определяемой по формуле

$$\lambda = \frac{n}{N_0 \cdot \Delta t} \quad \left[\frac{1}{\text{время}}\right],$$

где n — количество отказов, зафиксированное за время испытания Δt;

N_0 — количество испытуемых образцов.

Если назвать выражение $\frac{n}{N_0}$ считать относительным количеством отказов за время Δt, то **интенсивность отказов** можно определить как относительное количество отказов, приходящееся на каждую единицу времени.

Пример. Два автоматизированных рабочих места (АРМ) информационной системы испытывались 100 часов. В течение этого времени было зафиксировано 20 отказов. Интенсивность отказов составит:

$$\lambda = \frac{20}{2 \cdot 100} = 0{,}1 \quad \left[\frac{1}{\text{час}}\right].$$

Вероятность отказа $Q(T_0)$ — это вероятность появления отказа до конца заданного интервала T_0, т. е. это вероятность того, что наработка до первого отказа окажется меньше T_0.

Таким образом, понятие вероятности отказа соответствует функции распределения при экспоненциальном законе с параметром λ, равным интенсивности отказов[12]:

$$Q(T_0) = P(t < T_0) = F(T) = 1 - e^{-\lambda t}$$

Следовательно, плотность распределения вероятностей $f(T)$ определяется выражением

$$f(T) = \frac{d}{dt} F(T) = \lambda e^{-\lambda t}$$

В данном случае случайной величиной будет время T, а ее реализацией — наработки T_1, T_2 и т. д.

Вероятность безотказной работы $P(T_0)$ — это вероятность того, что объект сохранит работоспособность, т. е. не будет отказов в течение заданного интервала времени T_0. При этом не имеет значения, сколько проработал объект к началу временного интервала. Вероятность безотказной работы и вероятность отказа образуют полную группу несовместных и противоположных событий, и соответственно справедливо следующее:

$$P(T_0) = 1 - Q(T_0); \quad Q(T_0) = 1 - P(T_0).$$

Тогда их статистические оценки будут вычислены следующим образом:

$$Q(T_0) = \frac{n}{N_0}; \quad P(T_0) = 1 - \frac{n}{N_0},$$

где n — количество отказов за время испытаний;

N_0 — количество испытуемых объектов.

Пример. В течение 600 минут испытывались 10 АРМов системы. Зафиксировано 2 отказа.

Вероятность безотказной работы: $P(600)=1– 0,2 = 0,8$.

Вероятность отказа: $Q(600) = 2/10 = 0,2$.

При оценке надежности программного обеспечения в настоящее время достаточно широкое распространение нашла статистическая модель Миллса, которая основана на известном из теории вероятностей *гипергеометрическом законе распределения* и использовании метода максимального правдоподобия для нахождения наиболее вероятного значения неизвестной величины [3].

Статистическая модель оценки надежности программного обеспечения Миллса основана на следующей *технологии* [3]:

- программа «*засоряется»* некоторым количеством *известных, искусственных* ошибок. Эти ошибки вносятся в программу таким образом, чтобы смоделировать *случайную природу* ошибок, оставшихся в программе и, чтобы *вероятность* обнаружения внесенныых (искусственных) и собственных ошибок (все ещё находящихся в программе) при последующем тестировании была практически *одинакова* и зависела только от их количества
- проводится *тестирование* программы, в результате которого *обнаруживаются* как *внесенные*, так и *собственные* ошибки.

Пусть в программу внесены S ошибок (искусственных), и пусть при тестировании обнаружены $(n+v)$ ошибок, где n – найденные собственные ошибки, а v – найденные *внесенные* ошибки.

По методу *максимального правдоподобия* можно получить следующую оценку:

$N = S * n / v$, где:

N – первоначальное число собственных ошибок в программе.

Оценка для N может уточняться по мере обнаружения новых ошибок (по сравнению с ранее найденными) в программном обеспечении.

В статистической модели Миллса кроме оценки количества оставшихся в программе ошибок определяется также уровень достоверности этой оценки.

Задача решается в условиях предположения, что в программе имеется *не более*, чем K *собственных* ошибок.

Как и выше, в программу вносятся S искусственных ошибок.

После этого программа тестируется, пока не будут обнаружены *все* внесенные ошибки, при этом подсчитывается число обнаруженных сосбственных ошибок в программе. Пусть их окажется n.

Уровень значимости C может быть вычислен по формуле:

$$C = \begin{cases} 1 \text{ при } n > K \\ S/(S+K+1) \text{ при } n \le K \end{cases}, \text{ где}$$

C – вероятность того, что модель будет правильно отклонять ложное предположение.

Приведенные формулы для вычисления N и C образуют полную модель оценки надежности программного обеспечения. К недостаткам данной математической модели следует отнести необходимость проведения тестирования до тех пор, пока не будут обнаружены *все* внесенные ошибки.

К сожалению, применение подобных методов для определения априорной надежности программного обеспечения АСОИУ недостаточно эффективно, поскольку достоверность вычислений является довольно низкой. В реальных информационных системах при достижении требуемого уровня надежно-

сти, необходимого для нормального функционирования, отказы самих систем возникают из-за появления недокументированных (необнаруженных в ходе тестирования) ошибок. При этом рассчитываемые значения параметров надежности будут характеризоваться ***случайной ошибкой*** [3], значительно влияющей на эту величину. Однако подобные методы могут быть использованы для определения оценки показателей надежности аппаратуры, входящей в состав АСОИУ.

3.3. Методы повышения надежности АСОИУ

3.3.1. Минимизация влияния дестабилизирующих факторов

Для определения основных факторов, влияющих на нарушение надежности функционирования АСОИУ, т. е. дестабилизирующих факторов, следует определить составляющие части, (компоненты) системы, которые являются наиболее уязвимыми и влияют на надежность работы системы в целом. В [3] предлагается выделить следующие ***объекты уязвимости***:

- непосредственно вычислительный процесс обработки данных;
- информационную базу данных (БД) системы;
- откомпилированный программный код системы;
- информацию, выдаваемую пользователю в результате программной обработки.

Помимо приведенных объектов определенной уязвимостью обладает система управления базами данных (СУБД), от надежной работы которой зависит качество хранения информации в БД системы. Кроме того, если в качестве АСОИУ рассматривается информационно-аппаратная система, то одним из объектов уязвимости [3] является также аппаратное обеспечение. На выделенные объекты уязвимости влияют различные дестабилизирующие факторы. Принято выделять внутренние и внешние факторы. К ***внутренним факторам***, способным снизить надежность АСОИУ, можно отнести следующие дефекты [3]:

системные ошибки при постановке целей и задач создания АСОИУ (не следует путать с системными ошибками, возникающими в ходе функционирования программы). К этим ошибкам

можно отнести ошибки при формулировке требований к функциям и характеристикам решения задач, определении условий и параметров внешней среды;

алгоритмические ошибки, возникающие на стадии разработки системы (ошибки при определении структуры и взаимодействии компонент системы, неверное определение спецификации функций);

ошибки программирования в текстах программ и описаниях данных, а также ошибки в документации на систему;

недостаточную эффективность используемых методов и средств оперативной защиты программ и данных от различных программных и аппаратных сбоев и несанкционированного доступа.

К ***внешним факторам***, способным отрицательно влиять на определенные выше объекты уязвимости, в соответствии с [3] относятся следующие:

- ошибки оперативного и обслуживающего персонала в процессе эксплуатации ПС;
- искажения в каналах телекоммуникации информации, поступающей от внешних источников и передаваемой потребителям, а также недопустимые для конкретной информационной системы характеристики потоков внешней информации;
- сбои и отказы в аппаратуре вычислительных средств (этот фактор может относиться как к внешним, так и к внутренним факторам, если в состав АСОИУ входит аппаратное обеспечение);
- изменения состава и конфигурации комплекса взаимодействующей аппаратуры информационной системы, не отвечающие требованиям документации и не проверенные при испытаниях (тестировании) или сертификации.

Минимизация влияния перечисленных дестабилизирующих факторов может быть достигнута только при условии комплексного решения поставленных задач, что отражено в определенных методах обеспечения надежности АСОИУ, которые позволяют [3]:

- создавать высококачественные программные модули и функциональные компоненты;

• предотвращать дефекты проектирования за счет эффективных технологий и средств автоматизации обеспечения всего жизненного цикла комплексов программ и баз данных;

• обнаруживать и устранять различные дефекты и ошибки проектирования, разработки и сопровождения программ путем систематического тестирования на всех этапах жизненного цикла системы;

• удостоверять достигнутые уровни качества и надежности функционирования АСОИУ в процессе их испытаний и сертификации перед сдачей готового продукта в эксплуатацию;

• оперативно выявлять последствия дефектов программ и данных и восстанавливать нормальное, надежное функционирование системы.

3.3.2. Оптимизация процесса проектирования систем

Для обнаружения и устранения ошибок проектирования все этапы разработки и сопровождения системы должны быть поддержаны методами и средствами систематического тестирования. На этапах разработки АСОИУ целесообразно применять различные методы, эталоны и виды тестирования, каждый из которых ориентирован на обнаружение, локализацию или диагностику определенных типов дефектов. Надежность функционирования АСОИУ непосредственно зависит от полноты применяющихся комплектов тестов и адекватности генераторов тестов реальным объектам внешней среды и условиям будущей эксплуатации [3].

Кроме этого, свести к минимуму дефекты проектирования и значительно облегчить практическую реализацию ИС можно при помощи получивших широкое распространение CASE-систем и использования в качестве языка программирования одного из языков четвертого поколения 4GL. В широком понимании CASE-средства предназначены для автоматизации процессов проектирования информационных систем. Большинство существующих на рынке CASE-средств можно классифицировать по типам и категориям. Так, классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы жизненного цикла. Классификация по категориям определяет степень

интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи (tools), а также набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС (toolkit) и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием [13].

По типам CASE-средства можно классифицировать следующим образом [13]:

CASE-средства предназначенные для анализа и проектирования, результатом использования которых являются предварительные спецификации компонентов и интерфейсов информационной системы, алгоритмов и структур данных. К таким системам можно отнести Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA) и CASE.Аналитик (Макро-Проджект);

CASE-средства, назначение которых состоит в построении и анализе концептуальной модели предметной области, Design/IDEF (Meta Software), BPwin (Logic Works);

CASE-средства, предназначенные непосредственно для проектирования структуры баз данных на основе спроектированной концептуальной модели предметной области в идеологии конкретной СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE);

CASE-средства реинжиниринга — наиболее современные CASE-средства, предоставляющие возможности проведения широкомасштабного анализа программных кодов и схем баз данных и формирования на их основе различных моделей и предварительных проектных спецификаций. К таким Case-средствам можно отнести Rational Rose (Rational Software) и Object Team (Cayenne).

Рассмотрим наиболее часто встречающиеся на российском рынке CASE-средства, в функции которых входит создание концептуальной модели предметной области и физической структуры БД. При этом будем оперировать сведениями, представленными в [13].

CASE-средство Silverrun — разработка фирмы Computer Systems Advisers, Inc. (США). Silverrun ориентировано в боль-

шей степени на спиральную модель ЖЦ и может быть использовано для анализа и проектирования ИС бизнес-класса. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм «сущность-связь»).

Модуль концептуального моделирования данных ERX (Entity-Relationship eXpert) обеспечивает построение моделей данных «сущность-связь», не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему, позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.

Модуль реляционного моделирования RDM (Relational Data Modeler) позволяет создавать детализированные модели «сущность-связь», предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с построением базы данных: индексы, триггеры, хранимые процедуры и т. д. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы базы данных. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных баз данных.

Менеджер репозитория рабочей группы WRM (Workgroup Repository Manager) применяется как словарь данных для хранения общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.

Для автоматической генерации схем баз данных у Silverrun существуют средства интеграции с наиболее распространенными СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server,

SQLBase, Sybase. Это позволяет документировать, перепроектировать или переносить на новые платформы уже находящиеся в эксплуатации базы данных и прикладные системы. Таким образом, можно полностью определить ядро базы данных с использованием всех возможностей конкретной СУБД: триггеров, хранимых процедур, ограничений ссылочной целостности. Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных отчетов в соответствии с различными стандартами в системе Silverrun имеются различные способы выдачи.

Vantage Team Builder представляет собой интегрированный программный продукт, ориентированный на реализацию каскадной модели жизненного цикла программного обеспечения и поддержку полного ЖЦ ПО. Vantage Team Builder обеспечивает выполнение следующих функций:

- проектирование диаграмм потоков данных, ER-диаг-рамм, структур данных, структурных схем программ и последовательностей экранных форм;
- проектирование диаграмм архитектуры системы — SAD (проектирование состава и связи вычислительных средств, распределение задач системы между вычислительными средствами, моделирование отношений типа «клиент-сервер», анализ использования менеджеров транзакций и особенностей функционирования систем в реальном времени);
- генерацию кода программ на языке 4GL целевой СУБД с полным обеспечением программной среды и генерация SQL-кода для создания таблиц БД, индексов, ограничений целостности и хранимых процедур;
- программирование на языке С со встроенным SQL;
- управление версиями и конфигурацией проекта;
- многопользовательский доступ к репозиторию проекта;
- генерацию проектной документации по стандартным и индивидуальным шаблонам;
- экспорт и импорт данных проекта в формате CDIF (CASE Data Interchange Format).

При построении модели данных в виде ER-диаграммы в среде Vantage Team Builder выполняется ее нормализация и вводится определение физических имен элементов данных и таблиц, которые будут использоваться в процессе генерации физической схемы данных конкретной СУБД; кроме того, обеспечивается возможность определения альтернативных ключей сущностей и полей, составляющих дополнительные точки входа в таблицу (поля индексов), и мощности отношений между сущностями.

Uniface — продукт фирмы Compuware (США) — представляет собой среду разработки крупномасштабных приложений в архитектуре «клиент-сервер» и имеет в составе компонент Application Model Manager, который поддерживает прикладные ER-модели, каждая из которых представляет собой подмножество общей схемы БД с точки зрения данного приложения, и включает соответствующий графический редактор. В список поддерживаемых СУБД входят DB2, VSAM и IMS.

CASE-средство Designer/2000 фирмы ORACLE является интегрированным CASE-средством, обеспечивающим в совокупности со средствами разработки приложений Developer/2000 поддержку полного ЖЦ ПО для систем, использующих СУБД ORACLE.

Designer/2000 представляет собой семейство методологий и поддерживающих их программных продуктов. Базовой методологией Designer/2000 (CASE*Method) является структурная методология проектирования систем, полностью охватывающая все этапы жизненного цикла информационной системы.

В соответствии с этой методологией на этапе планирования определяются цели создания системы, приоритеты и ограничения, разрабатывается системная архитектура и план разработки ИС. В процессе анализа строятся модель информационных потребностей (диаграмма «сущность-связь»), диаграмма функциональной иерархии (на основе функциональной декомпозиции ИС), матрица перекрестных ссылок и диаграмма потоков данных.

На этапе проектирования разрабатывается подробная архитектура ИС, проектируется схема реляционной БД и программ-

ные модули, устанавливаются перекрестные ссылки между компонентами ИС для анализа их взаимного влияния и контроля за изменениями.

На этапе реализации создается БД, строятся прикладные системы, производится их тестирование, проверка качества и соответствия требованиям пользователей; создается системная документация; разрабатываются материалы для обучения и руководства пользователей. На этапах эксплуатации и сопровождения анализируются производительность и целостность системы, выполняется поддержка и при необходимости модификация ИС;

Designer/2000 обеспечивает графический интерфейс при разработке различных моделей (диаграмм) предметной области. В процессе построения моделей информация о них заносится в репозиторий.

Семейство продуктов ERWin от Logic Works предназначено для моделирования и создания баз данных произвольной сложности на основе диаграмм «сущность-связь». В настоящее время ERWin является наиболее популярным пакетом моделирования данных благодаря поддержке широкого спектра СУБД самых различных классов: SQL-серверов (Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Ingress, Rdb и др.) и «настольных» СУБД типа XBase (Clipper, dBASE, FoxPro, MS Access, Paradox и др.).

Информационная модель представляется в виде диаграмм «сущность-связь», отражающих основные объекты предметной области и связи между ними. Дополнительно определяются атрибуты сущностей, характеристики связей, индексы и бизнес-правила, описывающие ограничения и закономерности предметной области. После создания ER-диаграммы пакет автоматически генерирует SQL-код для создания таблиц, индексов и других объектов базы данных. По заданным бизнес-правилам формируются стандартные триггеры БД для поддержки целостности данных, для сложных бизнес-правил можно создавать собственные триггеры, используя библиотеку шаблонов.

Пакет позволяет осуществлять реинжиниринг существующих БД: по SQL-текстам автоматически генерируются ER-диаграммы. Таким образом, в пакете поддерживается техноло-

гия FRE (Forward and Reverse engineering), последовательность этапов которой приведена ниже:

1) импорт с сервера существующей БД;

2) автоматическая генерация модели БД;

3) модификация модели;

4) автоматическая генерация новой схемы и построение физической БД на том же самом или любом другом сервере.

Для разработки клиентской части приложения имеются специальные версии пакета, обеспечивающие интеграцию с такими инструментами, как SQLWindows, PowerBuilder, Visual Basic, Delphi. Предлагаются и усеченные версии продукта:

– ERWin/SQL, обеспечивающая лишь прямое проектирование для любых СУБД;

– ERWin/Desktop, поддерживающая технологию FRE только для «настольных» СУБД.

S-Designor представляет собой CASE-средство для проектирования реляционных баз данных. По своим функциональным возможностям и стоимости он близок к CASE-средству ERwin, отличаясь внешне используемой на диаграммах нотацией. S-Designor реализует стандартную методологию моделирования данных и генерирует описание БД для таких СУБД, как ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server и др. Для существующих систем выполняется реинжиниринг БД.

S-Designor совместим с рядом средств разработки приложений (PowerBuilder, Uniface, TeamWindows и др.) и позволяет экспортировать описание БД в репозитории данных средств. Для PowerBuilder выполняется также прямая генерация шаблонов приложений.

Visible Analyst Workbench фирмы Visible Systems представляет собой сетевое многопользовательское средство проектирования ИС, базирующееся на репозитории, хранимом на сервере SQLBase, Oracle или Informix. Пакет основан на методологии Мартина и поддерживает следующие диаграммные техники:

- диаграммы функциональной декомпозиции;
- диаграммы потоков данных в нотациях Йодана и Гейна-Сарсона;
- диаграммы «сущность-связь»;

• структурные карты в нотации Константайна.

Пакет обеспечивает генерацию схем БД для вышеперечисленных СУБД и поддерживает технологию FRE. Имеется возможность экспорта проектов в системы SQLWindows, PowerBuilder и Uniface. К достоинствам пакета может быть отнесено наличие развитых средств верификации проекта, и, прежде всего, возможностей вертикального и горизонтального балансирования диаграмм. Так, функциональная и информационная модели сильно коррелированы, что позволяет избавиться от лишних объектов моделей.

Power Designer компании Sybase — средство моделирования масштаба предприятия, объединяющего технологический уровень и бизнес-уровни моделирования для обеспечения максимально эффективного взаимодействия между бизнес- и IT-пользователями.

В состав Power Designer входят следующие модули:

Process Analyst — средство для функционального моделирования, поддерживающее нотацию Йордона–ДеМарко, Гейна–Сарсона и несколько других. Средством предоставляется возможность описания элементов данных (имен, типов, форматов), связанных с потоками данных и хранилищами данных. Эти элементы передаются на следующий этап проектирования, причем хранилища данных могут быть автоматически преобразованы в сущности;

Data Analyst (Architect) — инструмент для построения модели «сущность-связь» и автоматической генерации на ее основе реляционной структуры. Исходные данные для модели «сущность-связь» могут быть получены из DFD-моделей, созданных в модуле Process Analyst. В ER-диаграммах допускаются только бинарные связи, задание атрибутов у связей не поддерживается. Инструментом поддерживаются диалекты языка SQL примерно для 30-ти реляционных СУБД, при этом могут быть сгенерированы таблицы, представления, индексы, триггеры и т. д. В результате порождается SQL-сценарий (последовательность команд CREATE), выполнение которого создает спроектированную схему базы данных. Имеется также возможность установить соединение с СУБД через интерфейс ODBC. Другими воз-

можностями являются: автоматическая проверка правильности модели, расчет размера базы данных, реинжиниринг (построение модельных диаграмм для уже существующих баз данных) и т. д.;

Application Modeler — инструмент для автоматической генерации прототипов программ обработки данных на основе реляционных моделей, построенных в Data Analyst. С помощью данного инструмента может быть получен код для Visual Basic, Delphi, а также для таких систем разработки в архитектуре «клиент-сервер», как PowerBuilder, Uniface, Progress и др. Генерация кода осуществляется на основе шаблонов, соответственно управлять генерацией можно за счет изменения соответствующего шаблона.

Power Designer версии 9.5 и выше позволяет согласовывать объектно-ориентированную и концептуальную модели данных, ориентированную на реляционные СУБД.

Power Designer позволяет генерировать физическую структуру БД на основе спроектированной концептуальной модели для большинства современных СУБД (ORACLE, Informix, Ingres, Sybase, MS SQL Server и др.).

Design/IDEF фирмы Meta Software Corp. (Дания) автоматизирует все этапы проектирования сложных систем различного назначения: формулировку требований и целей проектирования, разработку спецификаций, определение компонентов и взаимодействий между ними, документирование проекта, проверку его полноты и непротиворечивости. Наиболее успешно пакет применяется для описания и анализа деятельности предприятия. Он позволяет оценить такую структуру, как единый организм, сочетающий управленческие, производственные и информационные процессы. В основе пакета лежит методология структурного проектирования и анализа сложных систем IDEF0/SADT.

Design/IDEF строит иерархические модели сложных систем посредством декомпозиции ее компонентов; поддерживает коллективную разработку IDEF-модели, позволяя в любой момент объединять различные подмодели в единую модель системы; создает словарь данных для хранения информации о функциях и структурах данных проекта; формирует 5 типов отчетов, поддерживающих процесс разработки и анализа моделей.

Disign/IDEF реализован на платформах MS Windows, Macintosh Plus и выше, Sun Solaris (X Window System), HP9000 модели 700 и 800 (X Window System). Disign/IDEF также интегрирован с пакетом динамического анализа сложных систем WorkFlow Analyzer и пакетом функционально-стоимостного анализа EasyABC.

3.3.3. Проверка достоверности надежности АСОИУ

Для удостоверения качества, надежности и безопасности применения АСОИУ используемые в них подсистемы следует подвергать обязательной сертификации и аттестованным, проблемно-ориентированным испытаниям {2]. Такие испытания необходимо проводить, когда создаваемые программы предназначаются для управления сложными процессами или обработки настолько важной информации, что дефекты в них или недостаточное качество могут нанести значительный ущерб.

Сертификационные испытания должны устанавливать соответствие комплексов программ документации к системе 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, независимыми от разработчиков и от заказчиков (пользователей).

Испытания АСОИУ 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 завершающих сертификационных испытаниях [3]. При успешном проведении испытаний на каждый программный продукт выдается сертификат, подтверждающий соответствие созданной системы нормативной документации, а также возможность использования системы в определенной предметной области.

К основным оперативным методам, повышающим надежность АСОИУ, можно отнести использование средств поддержки целостности БД, а также средств восстановления системы после различных программных и аппаратных сбоев.

3.4. Избыточность как средство повышения надежности АСОИУ

Избыточность АСОИУ характеризуется наличием в системе дополнительных возможностей сверх тех, которые могли бы обеспечить ее нормальное функционирование. Избыточность вводится для повышения надёжности работы системы в различных условиях эксплуатации или для исключения влияния на достоверность передаваемой информации помех и сбоев, возникающих в передающей (приёмной) аппаратуре. Частным видом избыточности является информационная избыточность.

Основным из требований, предъявляемым к надежности АСОИУ, является надежность хранения информации в базе данных системы. Для выполнения требования надежного хранения БД необходимо поддерживать избыточность хранения данных, что обычно реализуется в виде журнала изменений БД.

Главным критерием при выборе СУБД, под управлением которой будут храниться данные системы, является ***надежность хранения информации*** — возможность восстановления системой управления базой данных последнего согласованного состояния БД после какого-либо сбоя.

Различают два вида сбоев:

1) аппаратные сбои, разделяющиеся на мягкие сбои, возникающие, например, при аварийном выключении питания, и жесткие сбои, при которых возможна частичная или полная потеря информации;

2) программные сбои — аварийное завершение работы СУБД.

В результате программных и аппаратных сбоев во время работы пользователя с БД некоторые транзакции могут остаться незавершенными (*транзакция* — последовательность операций над БД, рассматриваемых СУБД как единое целое.). Для восстановления БД необходимо хранить информацию об изменениях, производимых в БД, т.е. поддерживать журнал изменений БД.

Журнал — это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физи-

ческих дисках). В журнал поступают записи обо всех изменениях основной части БД [5]. Идеология формирования журнала основывается на необходимости соблюдения принципа упреждающей записи об изменениях БД в журнал WAL (Write Ahead Log), т. е. информация об изменениях данных в БД в журнале должна появиться до того, как произойдут эти изменения. Если в СУБД корректно ведется журнал БД, то с его помощью можно восстановить базу после любого сбоя (естественно, если в результате сбоя не утерян сам журнал).

Основным способом восстановления БД является индивидуальный откат транзакций.

При мягком сбое в БД могут находиться данные, модифицированные транзакциями, не закончившимися к моменту сбоя, и могут отсутствовать данные, модифицированные транзакциями, которые к моменту сбоя успешно завершились. Целью процесса восстановления после мягкого сбоя является достижение состояния внешней памяти основной части БД, которое возникло бы при фиксации во внешней памяти изменений всех завершившихся транзакций и которое не содержало бы никаких следов незаконченных транзакций [5]. Процесс восстановления производится путем отката незавершенных транзакций, после чего повторно выполняются операции завершенных транзакций, результаты которых не отражены в БД.

При жестком сбое для восстановления БД необходимо использовать кроме журнала и архивную копию БД, находящейся в согласованном состоянии. Принцип восстановления БД состоит в том, что по архивной копии и, следуя журналу, должны быть отработаны все завершенные транзакции.

Информационная избыточность может быть обеспечена также путем дублирования всей информации, расположенной на жестком диске, методом зеркалирования дисков.

Кроме этого, в системе необходимо обеспечить наличие дополнительных конструкций, обеспечивающих защиту от несанкционированного доступа (более подробно см. подраздел 2.3).

В том случае, если в состав АСОИУ входит специальное аппаратное обеспечение, в системе необходимо обеспечить наличие средств контроля аппаратуры, а также дополнительных

устройств, обеспечивающих ее помехоустойчивость и помехозащищенность.

Применяются и другие виды избыточности: аппаратурная избыточность; временная избыточность, т. е. запас времени для повторного выполнения операции (например, проведения двойного или тройного расчета одной и той же функции); энергетическая избыточность — запас мощностей, который может быть использован в более тяжёлых условиях эксплуатации (например, установка более мощного двигателя, чем это необходимо в нормальных условиях его работы).

Контрольные вопросы

1. Дайте определение понятия надежности.
2. Сформулируйте задачи теории и анализа надежности АСОИУ.
3. Перечислите объекты уязвимости АСОИУ
4. Определите внутренние и внешние дестабилизирующие факторы, способные снизить надежность.
5. Дайте характеристику методов обеспечения надежности АСОИУ.

4. ЭРГОНОМИКА АСОИУ

4.1. Общие сведения

Одним из важных понятий качества АСОИУ с точки зрения удобства использования является эргономичность системы. **Эргономика** (от греч. ergon — работа и nomos — закон) — отрасль науки, изучающая человека (или группу людей) и его (их) деятельность в условиях производства с целью совершенствования орудий, условий и процесса труда. Основной объект исследования эргономики — системы «человек-машина». Эргономика изучает движение человека в процессе производственной деятельности, затраты его энергии, производительность и интенсивность при конкретных видах работ.

Эргономика подразделяется на миниэргономику, мидиэргономику и макроэргономику. В основу эргономики легли многие дисциплины от анатомии до психологии, а главной ее задачей является создание таких условий работы для человека, которые бы способствовали сохранению здоровья, повышению эффективности труда, снижению утомляемости, да и просто поддержанию хорошего настроения в течение всего рабочего дня.

Термин «эргономика» был принят в Англии в 1949 г. при создании группой английских ученых Эргономического исследовательского общества. В СССР в 1920-е годы предлагался термин «эргология», а в настоящее время принят английский термин. В некоторых странах эта научная дисциплина имеет иные названия: в США — «исследование человеческих факторов» (Human Factors); в ФРГ — «антропотехника».

Эргономика претерпела существенные изменения в процессе своего развития. Так, если 20 лет назад основные работы велись в области антропометрии, физиологии труда, проектирования труда, биомеханики, психологии, то в последнее десятилетие приоритеты эргономики существенно сместились в область безопасности, проектирования труда, биомеханики, напряженности труда, интерфейса «человек-компьютер». Биомеханика и физиология труда не доминируют, как в прошлом, но возник их новый аспект, связанный с расстройствами опорно-двигательного аппа-

рата, обусловленный увеличением числа людей, работающих на компьютеризированных местах.

В литературе развитие эргономики по десятилетиям характеризуется следующим образом [14]:

1950-е годы — военная эргономика;

1960-е годы — промышленная эргономика;

1970-е годы — эргономика товаров широкого потребления;

1980-е годы — интерфейс «человек-компьютер» и эргономика программного обеспечения;

1990-е годы — когнитивная и организационная эргономика.

Эргономика является одновременно и исследовательской и проектировочной дисциплиной, так как одной из её задач является разработка методов учета человеческих факторов при проектировании новой и модернизации старой техники и технологии, а также существующих условий труда. Объектом исследования эргономики является система «человек — машина — среда» (СЧМ). Эргономика рассматривает СЧМ как сложное функционирующее целое, в котором ведущая роль принадлежит человеку. Человека, работающего с помощью машины, принято называть оператором.

Эргономика рассматривает технический и человеческий аспекты в неразрывной связи. Сочетание способностей человека и возможностей машины существенно повышает эффективность функционирования СЧМ. Поэтому решение прикладных проблем эргономики предполагает движение одновременно в двух направлениях — от требований человека к машине и условиям ее функционирования и, наоборот — от требований машины и условий ее функционирования к человеку. Оптимальные решения находятся, как правило, на пересечении этих направлений. Тем самым эргономика решает задачи рациональной организации деятельности людей в СЧМ, целесообразного распределения функций между человеком и машиной.

Следует особо подчеркнуть, что эргономика изучает определенные свойства СЧМ, получившие название ***человеческих факторов***. Они представляют собой интегральные характеристики связи человека и машины, проявляющиеся в конкретных условиях их взаимодействия при функционировании системы.

Знание человеческих факторов позволяет формулировать требования к профессиональному отбору и обучению персонала, техническим средствам подготовки, согласованию внешних средств трудовой деятельности и способов ее осуществления. Увеличивается роль человеческих факторов применительно к задачам проектирования, создания и использования технически сложных изделий культурно-бытового назначения (радиоаппаратуры, магнитофонов, телевизионной техники и др.). Важным фактором, например, является эргономика рабочего места пользователя при работе с компьютером. По данным Министерства труда США так называемые «повторяющиеся травмирующие воздействия при работе с компьютером» (ПТВРК) обходятся корпорациям Америки ежегодно в 100 млрд долл. Компенсации, выплачиваемые их служащим, достигают астрономических цифр. Кроме того, некоторым пострадавшим от работы за ПК приходится расплачиваться жестокими болями в течение всей жизни. Чтобы снизить влияние таких нагрузок на организм человека, необходимы «эргономично спроектированные рабочие места и правильные навыки работы с компьютером».

В качестве практически идеального рабочего места можно представить себе некую угловую структуру с передним краем в форме лекала. Общая занимаемая ею площадь оптимальна, если ее установить в углу, а «мертвое», недоступное пространство занять монитором. Его сглаженный полукруглый передний край образует дугу вокруг пользователя, обеспечивая оптимальную зону доступа к компьютеру.

Человеческие факторы всесторонне проявляются и фиксируются в такой целостной эргономической характеристике СЧМ, как эргономичность. **Эргономичность** — свойство техники изменять эффективность трудовой деятельности в СЧМ в зависимости от степени ее соответствия физическим, биологическим и психическим свойствам человека. Эргономичность формируется на базе таких свойств техники, как управляемость, обслуживаемость, осваиваемость и обитаемость.

Управляемость — свойство техники изменять эффективность выполнения человеком основной и вспомогательной рабо-

ты при обеспечении необходимых технологических операций над предметом труда.

Обслуживаемость — свойство техники изменять эффективность выполнения человеком трудовых операций по приведению техники в состояние готовности к функционированию и поддержанию этого состояния во времени.

Осваиваемость характеризует эффективность приспособления техники к быстрому и качественному овладению ею обслуживающим и управляющим персоналом.

Обитаемость — эргономическое свойство техники, приближающее условия её функционирования к оптимальным биологическим параметрам внешней среды, при которых работающему человеку обеспечивается нормальное развитие, хорошее здоровье и высокая работоспособность.

Качественными показателями эргономичности являются:

1) в области управляемости:

- среднее время или коэффициент занятости человека-оператора выполнением определенной единицы технологического процесса;
- вероятность выполнения человеком-оператором единицы технологического процесса с заданным качеством;
- производительность (норма времени на единицу труда);

2) по уровню обслуживаемости:

- среднее оперативное время занятия человека подготовкой техники к её применению;
- среднее оперативное время занятостью восстановлением или профилактикой техники;

3) по степени осваиваемости:

- среднее календарное время профессиональной подготовки человека-оператора;
- уровень квалификации человека, необходимый для обслуживания техники.

4.2. Оптимальные задачи эргономики

Одной из важнейших задач эргономики является оптимизация условий труда путем оптимально-рационального проекти-

рования оборудования рабочих мест с учетом возможностей и особенностей различных категорий индивидов. Эргономика приобретает все большее значение и в решении комплексной проблемы реабилитации лиц, в той или иной мере утративших работоспособность. С этой целью в эргономике изучаются психофизические возможности и особенности людей пожилого возраста. Таким образом, эргономика создает научную базу для решения важной социальной проблемы по вовлечению в производительный труд указанной части населения.

Эргономика призвана решать ряд проблем, связанных с оценкой точности, надежности и стабильности работы, влияния психической напряженности, утомляемости, эмоциональных факторов и особенностей нервно-психической организации оператора на эффективность его деятельности в СЧМ. Большое значение имеет создание эргономического обеспечения научной организации и безопасных условий труда. С этой целью должна производиться разработка эргономических норм и требований, а также эргономическая оценка качества промышленной продукции, в том числе и информационных систем.

Решение любой эргономической задачи состоит из последовательности этапов:

1) анализ деятельности оператора в СЧМ, в процессе которого изучается структура его деятельности, выявляются цели, мотивы и способы выполнения трудовых действий, рассматриваются возможные режимы работы и оценивается их влияние на результаты труда. На основании этих исследований определяются необходимые требования к характеристикам человека-оператора;

2) изучение комплекса эргономических свойств (характеристик) человека-оператора: исследование работы органов чувств человека, его центральной нервной системы, моторно-двигательного аппарата и т. д. Причем рассматриваются только оптимальные значения этих характеристик, а не экстремальные;

3) организация рабочего места оператора с учетом его эргономических свойств. Разработка требований, предъявляемых к рабочему месту в целом и отдельным его элементам, с целью обеспечения максимальных удобств и эффективности работы;

4) организация профессиональной подготовки операторов, включающей в себя профотбор, профобучение, тренировку и формирование коллективов;

5) эргономическое проектирование и оценка СЧМ;

6) определение экономического эффекта эргономического обеспечения.

Поскольку деятельность человека-оператора является основным предметом эргономического исследования, рассмотрим его психофизиологическую сущность более подробно.

Наиболее характерной чертой деятельности оператора является отсутствие возможности непосредственно наблюдать за управляемыми объектами и необходимость использования информации, которая поступает к нему по каналам связи. Деятельность человека, совершаемую не с реальными объектами, а с их заместителями или имитирующими их образами, называют деятельностью с информационными моделями реальных объектов.

Информационная модель — совокупность информации о состоянии и функционировании объекта управления и внешней среды. Модель является для оператора своеобразным имитатором, отражающим все существенно важные для управления свойства реальных объектов. Модель является источником информации, на основе которого формируется образ реальной обстановки, производится анализ и оценка сложившейся ситуации, планируются управляющие воздействия, принимаются решения, обеспечивающие правильную работу системы и выполнение возложенных на нее задач, а также наблюдает и оценивает результаты их реализации. Объем информации, включенной в модель, и правила ее организации должны соответствовать задачам и способам управления. Физически информационная модель реализуется с помощью устройств отображения информации. Наиболее существенной особенностью в деятельности человека с такой моделью является необходимость соотнесения сведений, получаемых с помощью приборов, экранов, табло, как между собой, так и с реальными управляемыми объектами. Именно на основании соотнесения этих сведений строится вся деятельность оператора.

Рассмотрим основные этапы деятельности оператора при решении определенной технологической задачи или выполнении операции СЧМ:

1) восприятие информации — процесс, включающий качественно различные операции: обнаружение объекта восприятия; выделение в объекте отдельных признаков, отвечающих стоящей перед оператором задаче; ознакомление с выделенными признаками и опознавание объекта восприятия;

2) оценка информации, ее анализ и обобщение на основе заранее заданных или сформированных критериев оценки. Оценка производится на основе сопоставления воспринятой информационной модели со сложившейся у оператора внутренней образно-концептуальной моделью обстановки (системы управления).

Перейдем к описанию эргономических проблем, с которыми сталкиваются пользователи и разработчики АСОИУ.

4.3. Основные эргономические проблемы АСОИУ

Эргономика программного обеспечения — это подраздел микроэргономики, ориентированный на системы «человек — компьютер», «человек — компьютер — человек», «человек — компьютер — процесс», «человек — программа» и т. п. Человек является составной частью любой программно-информационной системы, ориентированной на использование в любых процессах его жизнедеятельности. При этом необходимо отметить следующее: как программная система способна прямо или косвенно влиять на человека, так и человек способен каким-либо образом влиять на работу системы. Такая связь человека с информационной системой должна находиться в согласованном (гармоничном) состоянии. Место и роль человека сводится к восприятию, оценке информации, поступающей из разнородных источников, принятию решений, формированию и реализации команд для исполнения. Человек осуществляет также контроль состояния самой системы.

Выделяют три типа эргономических проблем, возникающих при разработке и использовании АСОИУ [14]:

1) глобальные эргономические противоречия;

2) неадекватное применение интерфейсной парадигмы;

3) ошибки проектирования элементов пользовательского интерфейса (как целых форм, так и аспектов применения конкретных элементов управления).

В основе эргономических требований к создаваемым АСОИУ лежит требование к эргономичности пользовательского интерфейса. При разработке системы создается описание внешнего дизайна программного продукта, которое включает полное описание его пользовательского интерфейса, в частности все экранные и печатные формы. Иногда, если работа выполняется для конкретного заказчика, внешний дизайн программного продукта могут определять его пользователи, создавая часть проектных документов в тех терминах, которые им понятны.

В процессе разработки продукта его внешний дизайн может многократно изменяться, поскольку при кажущейся второстепенности именно эта часть системы имеет ключевое значение для общего восприятия системы пользователем. Естественно, что пользователя не удовлетворит безупречность программного кода продукта, если какая-то часть интерфейса вызывает у него затруднения (запутывает, ведет к ошибкам, раздражает или является недостаточно гибкой и функциональной) т. е. не делает всего того, что, по мнению пользователя, она обязательно должна уметь. Однако, работая над внешним дизайном, необходимо понимать, что даже в наиболее тщательно продуманной системе в процессе ее эксплуатации все равно могут обнаружиться некоторые недостатки.

Более подробно остановимся на значении пользовательского интерфейса автоматизированных информационных систем и организации взаимодействия пользователя с системой.

4.4. Эргономика пользовательского интерфейса АСОИУ

4.4.1. Основные принципы проектирования

Проектирование пользовательского интерфейса — это работа в основном по анализу деятельности реальных или потенциальных пользователей продукта: их желаний, стремлений, предпочтений и особенностей работы [15].

Цель создания эргономичного пользовательского интерфейса заключается в том, чтобы отобразить информацию достаточно эффективно, насколько это возможно для человеческого восприятия, и структурировать отображение на мониторе таким образом, чтобы привлечь внимание к наиболее важным единицам информации. При этом следует минимизировать общую информацию на экране и представить только те компоненты системы, которые необходимы для пользователя в конкретной ситуации.

Определим основные принципы создания эргономичного интерфейса системы:

***естественность* (*интуитивность*).** Работа с системой не должна вызывать у пользователя сложностей в поиске необходимых элементов интерфейса для управления процессом решения поставленной задачи;

непротиворечивость. Если в процессе работы с системой пользователем были использованы некоторые приемы работы с определенной частью системы, то в другой части системы приемы работы должны быть идентичны (однотипны). Работа с системой посредством интерфейса должна соответствовать установленным нормам (например, использование клавиш Enter, Delete и т. д.);

неизбыточность. Пользователь должен вводить только минимальную информацию для работы или управления системой. Например, пользователь не должен вводить незначимые цифры (00010 вместо 10). Аналогично, нельзя требовать от пользователя ввести информацию, которая была предварительно введена, или информацию, которая может быть автоматически получена из системы. Желательно использовать значения по умолчанию там, где только это возможно, с целью минимизации процесс ввода информации;

***непосредственный доступ к системе помощи*.** В процессе работы необходимо, чтобы в системе был обеспечен доступ пользователя к требуемым инструкциям. Система помощи должна описывать оптимальное количество существующих команд на должном уровне и обеспечивать пояснения характера сообщений об ошибках и пояснение выполняемого системой действия. Сообщения об ошибках должны быть полезны и понятны пользователю;

гибкость — способность интерфейса системы обслуживать пользователей с различными уровнями подготовки. Для неопытных пользователей интерфейс может быть организован как иерархическая структура меню, а для опытных пользователей — как команды, комбинации нажатий клавиш и параметры.

Встраивание задач эргономического проектирования в общий процесс разработки системы — чрезвычайно важный момент создания программных продуктов. Особенно это существенно при разработке приложений для компьютерных сетей и Интернет.

Качественно организованный пользовательский интерфейс позволяет реализовать следующие преимущества:

- снижение количества производимых пользователем ошибок при работе с системой;
- снижение стоимости поддержки системы;
- снижение стоимости обучения;
- уменьшение потерь вследствие снижения производительности работников при внедрении системы и быстрое восстановление прежнего уровня;
- улучшение морального состояния персонала;
- уменьшение расходов на изменение дизайна пользовательского интерфейса по требованию пользователей;
- доступность функциональности системы для максимального количества пользователей.

Ориентированные на пользователей методы проектирования пользовательского интерфейса демонстрируют определенные преимущества. Очевидно, что идентификация и устранение ошибок на более раннем этапе проектирования системы ведет к ее значительному удешевлению. Например, такие методы, как бумажное макетирование пользовательского интерфейса совместно с конечными пользователями, ведет к установлению более полного понимания между заказчиком и разработчиком ПО, что, в свою очередь, снижает вероятность последующих переделок.

Более полное и четкое определение задач (не только с точки зрения технологий, но и с точки зрения будущих пользователей системы) и договоренность относительно принципов построения пользовательского интерфейса ведет к более адекватной

оценке задачи, как заказчиком, так и исполнителем, позволяет заказчику убедиться в том, что исполнитель действительно заботится о его потребностях. Приведем основные рекомендации по созданию дружественного пользовательского интерфейса.

4.4.2. Размещение информации на экране

Количество информации, отображаемой на экране, называется ***экранной плотностью***. Исследования показали: чем меньше экранная плотность, тем отображаемая информация наиболее доступна и понятна для пользователя и наоборот, большая плотность может вызвать затруднения в восприятии и усвоении информации и ее понимании. Однако некоторые опытные пользователи предпочитают интерфейсы с большой экранной плотностью.

Информация на экране может быть сгруппирована и упорядочена в значимые части. Это может быть достигнуто с использованием кадров (фреймов), методов цветового кодирования, рамок, негативного изображения или других методов для привлечения внимания пользователей системы.

Имеется несколько полезных правил относительно расположения информации на экране [16]:

1) данные должны располагаться так, чтобы пользователь мог просматривать их в логической последовательности, т. е. таким образом чтобы направление просмотра формировалось из левого верхнего угла слева направо и сверху вниз;

2) данные должны располагаться таким образом, чтобы пользователь мог идентифицировать связанные группы информации. Отдельные группы логически связанных данных можно отделять вертикальными и горизонтальными линиями, помещать в отдельные ниши, панели, однако при этом не следует перегружать форму лишними графическими элементами;

3) информация должна располагаться таким образом, чтобы окно было композиционно «уравновешенным», т. е. «центр тяжести информации» должен быть примерно посредине окна. Желательно также, чтобы информация не была слишком плотной, чтобы не утомлять пользователя;

4) расположение одинаковой или сходной информации в различных окнах должно быть согласованно. Желательно использование единого шаблона. Во время проектирования изображений полезно нарисовать их на разлинованной бумаге. При этом элементы, являющиеся общими для различных изображений (например, кнопки Ok, Cancel), следует помещать в одно место.

4.4.3. Выделение элементов интерфейса

Для привлечения внимания к каким-либо элементам интерфейса можно воспользоваться выделением этих элементов большей яркостью на фоне других, более темных. Однако необходимо учесть, что большое количество ярких элементов может вызвать дискомфорт у пользователя. Таким образом, можно достичь обратного эффекта — перегрузки интерфейса. Применять этот метод нужно только при необходимости. Существует несколько способов выделения яркостью:

движение (мигание или изменение позиции). Очень эффективный метод, поскольку глаз имеет специальный детектор для движущихся элементов;

увеличение яркости элементов. Не очень эффективный метод, поскольку большинство людей могут обнаружить всего лишь несколько уровней яркости;

цвет. Использование цвета может быть чрезвычайно эффективно;

форма (символ, шрифт, форма символа). Используется для того, чтобы отличить различные категории данных;

различные алфавиты (шрифты) в разных формах;

размер (текста, символов). Обычно применяют увеличение выделенного объекта в 1,5 раза;

оттенение (различная текстура объектов). Эффективный метод для привлечения внимания к какой-либо части экрана;

окружение (подчеркивание, рамки, инвертированное изображение). Очень эффективный метод привлечения внимания пользователя системы.

Цвет может улучшить интерфейс пользователя, но для многих систем использование цвета практически не влияет на эффективность работы пользователя. Основное назначение цвета

— создание интерфейсов, более удобных для пользователей. Имеются случаи, где цвет может помочь проектировщику интерфейса пользователя. Наиболее эффективно использование варьирования цветовых характеристик в следующих ситуациях:

- группировка информации;
- выделение различий между информацией;
- выделение простых сообщений (ошибки, состояния и т. д.)

Цвет — мощный визуальный инструмент, который необходимо использовать очень осторожно, чтобы не вызвать дискомфорта у пользователя цветовыми комбинациями. Выбор цвета и цветовых сочетаний не должен быть хаотичным. Цвет, будучи правильно приложен, может грандиозно обогатить интерфейс пользователя, улучшая его эстетические качества, и привлекая внимание пользователя к важным элементам системы. Неверное использование цвета, с другой стороны, может серьезно повредить способности пользователя гармонично взаимодействовать с программой.

Ниже представлены принципы использования цвета, которыми рекомендуется руководствоваться при проектировании действительно эргономичного интерфейса:

- используйте минимальное количество цветов (не более 3–4), поскольку слишком пестрые изображения способны быстро утомить глаза; для неактивных элементов нужно использовать бледные цвета;
- если цвет используется для кодировки информации, необходимо удостовериться, что пользователь правильно понимает код, например, просроченные счета в бухгалтерских системах выделяются красным цветом, а непросроченные — зеленым;
- необходимо использовать цвета согласно представлениям пользователя (например, для картографа зеленый цвет обозначает лес, желтый — пустыню, синий — воду; для химика, красный — горячий, синий — холодный);
- для отображения состояния цвета могут быть использованы следующим образом: красный — опасность либо остановка программы, зеленый — нормальное состояние (продолжение работы), желтый — предостережение;
- для привлечения внимания пользователя к событию или

объекту наиболее эффективны белый, желтый и красный цвета. При этом опять же необходимо учитывать общепринятые представления о цветах (поскольку красный цвет считается цветом опасности, его лучше использовать в сообщениях об ошибках).

- для разделения данных необходимо выбрать цвета из различных частей спектра (красный — зеленый, синий — желтый, любой цвет — белый);
- текст и изображение должны четко выделяться на фоне. Например, нельзя использовать желтый цвет на белом фоне, а синий — на черном;
- для группировки данных, объединения и подобия нужно использовать оттенки цвета, которые являются соседями в спектре (оранжевые — желтые, синие — фиолетовые);
- для фона лучше использовать более спокойные тона. Если в изображении используется большое количество цветов, фон лучше сделать белым или серым. На светлом фоне цвета кажутся ярче и способны значительно легче восприниматься при различном внешнем освещении;
- некоторые комбинации цветов или оттенков могут быть неприятны для глаз, например, голубой цвет символов на красном фоне способен вызвать раздражение пользователя.

Кроме того, необходимо отметить, что около 9 % людей не различают цвета (обычно красно-зеленые сочетания). Однако эти люди могут отличать черно-белые оттенки, поэтому специалисты по разработке пользовательского интерфейса автоматизированных систем должны проверять, не нарушает ли восприятие пользователей этой категории использование различных цветов в экранных формах программных продуктов.

4.4.4. Непротиворечивость и стандартизация

Данные на экране следует располагать таким образом, чтобы пользователь знал, где их найти и в какой части экрана ожидать вывода необходимой информации. При этом необходимо соблюдать следующие правила:

- информация, на которую следует немедленно обратить внимание, должна всегда отображаться на самом видном месте, чтобы захватить внимание пользователя (например, предупре-

ждающие сообщения и сообщения об ошибках целесообразней размещать в центре экрана);

- информация, которая редко используется (например, справка) не должна отображаться постоянно, но должна быть доступна, когда потребуется. Например, иконка справки или соответствующая опция меню должна быть доступна на каждом экране;
- менее срочная или менее необходимая информация не должна все время находиться перед пользователем, но также должна быть доступна, когда понадобится (например, при использовании поля со списком рекомендуется обеспечить открытие самого списка только в случае ввода информации в это поле — использовать выпадающий список combo box);
- гиперссылки и ссылки на объекты системы должны быть сгруппированы по алфавиту.

4.4.5. Меню и пиктограммы (иконки)

Важной частью пользовательского приложения является меню, позволяющее пользователю выполнять задачи внутри приложения и управлять информационным процессом. **Меню приложения** — это набор опций, отображаемых на экране, с помощью которых пользователи могут выбирать и выполнять действия, тем самым производя изменения в состоянии интерфейса и управлять системой.

Достоинство меню в том, что пользователи не должны запоминать название элемента или действия, которое они хотят выполнить, они должны только распознать его среди пунктов меню. Таким образом, меню может использовать даже неопытный пользователь. Однако проект меню должен быть тщательно продуман: для того чтобы меню стало по-настоящему эффективным, названия пунктов меню должны быть очевидными.

Если АСОИУ относится к классу сложных систем, меню пользовательского приложения может занимать достаточно много экранного места. Чтобы избежать подобной ситуации целесообразно использовать так называемое всплывающее или ниспадающее меню, при этом нажатие на строку меню вызывает соответствующее всплывающее или ниспадающее подменю.

В процессе проектирования меню приложения необходимо принять наилучший способ отображения меню, чтобы оно было понятно и легко в использовании. Обычно команды меню упорядочены некоторым иерархическим способом. Основная проблема состоит в том, чтобы правильно распределить различные пункты меню по различным уровням и правильно их сгруппировать.

Исследования показывают, что имеются четыре варианта для организации меню пользовательского приложения:

1) алфавитный;

2) категорийный;

3) в соответствии с принятыми соглашениями;

4) в соответствии с частотой использования.

Принципы проектирования меню можно определить следующим образом:

- структура меню должна соответствовать структуре решаемой системой задачи, организация меню должна отражать наиболее эффективную последовательность шагов для достижения поставленной цели;
- пункты меню должны быть краткими, грамматически правильными и соответствовать своему заголовку в меню. Порядок пунктов меню выбирается согласно соглашению, частоте использования, порядку использования, в зависимости от потребностей задачи или пользователя;
- выбор пунктов меню должен быть обеспечен несколькими способами: с помощью клавиатуры, с помощью мыши, а также через другие объекты пользовательского интерфейса.

Необходимо использовать легко запоминаемые сочетания клавиш для более быстрого доступа к пунктам меню (например, нажатие сочетания клавиш Ctrl-S во многих системах аналогично вызову пункта меню «Файл — Сохранить»), что способно значительно сэкономить общее время выполнения процессов пользователями в системе.

Другими важными элементами информационной системы являются графические **иконки** (**пиктограммы**). Все иконки можно классифицировать согласно тому, насколько они отображают несущую функцию:

• иконки подобия — иконки похожие на объекты, которые они отображают (например, иконка с изображением ножниц может служить для отображения операции «вырезки» фрагмента текста или какого либо объекта);

• иконки по образцу представляют пример типа объекта (например, иконка, показывающая линию, может служить для вызова средства рисования линии);

• символические иконки используются для представления действия или состояния в символической форме (например, разорванная линия между двумя компьютерами может обозначать разорванное сетевое соединение);

• произвольные иконки не несут никакой информации по поводу их представления, поэтому их назначение должно быть описано (например, обратная круговая стрелка в большинстве случаев обозначает действие «отмена последней команды»).

4.4.6. Формы

Форма — основной элемент интерфейса. Назначение формы — удобный ввод и просмотр данных, состояния и сообщений автоматизированной системы.

Форма должна быть сформирована в соответствии с принципом «минимального объема памяти пользователя». Это говорит о том, что пользователю не должна выдаваться лишняя, избыточная информация, не связанная с текущим шагом решения задачи. Лишняя информация отвлекает и утомляет пользователя. Пояснения лучше поместить в справочную систему.

С другой стороны, важно, чтобы пользователю была предоставлена вся необходимая информация. При этом необходимо учитывать, что он не должен запоминать ранее предоставленную информацию, чтобы воспользоваться ею на данном шаге. Например, если выдается запрос «Вы желаете сохранить данные в файле?», следует уточнить имя файла.

Основные принципы проектирования формы:

• форма проектируется с целью обеспечения удобного, понятного и быстрого способа достижения решения поставленной в приложении задачи. Если экранная форма проектируется на основе бумажной формы, то передвижение по смежным полям

не должно вызывать затруднений у пользователя;

- размещение информационных единиц на пространстве формы должно соответствовать логике ее будущего использования. Это зависит от необходимой последовательности доступа к информационным единицам, частоты их использования, а также от относительной важности элементов;
- важно использовать в формах незаполненное пространство для создания равновесия и симметрии среди информационных элементов формы и фиксации внимания пользователя;
- логические группы элементов необходимо отделять пробелами, строками, цветовыми или другими визуальными средствами, стараясь при этом не перегружать форму лишними элементами, способными нарушить целостное восприятие информации;
- взаимозависимые или взаимосвязанные элементы информационных данных должны отображаться в одной форме.

В многооконном многозадачном графическом интерфейсе существует возможность менять фокус ввода, переключаться в другие окна, одновременно работать с несколькими объектами. Однако бывают ситуации, когда по логике процесса работать с другим объектом нельзя до тех пор, пока не будет завершена обработка сообщения, не будет введен пользователем ответ на запрос программы, не будет устранена неполадка и так далее. Активное окно называется ***модальным***, когда переход к другому окну невозможен без его закрытия. Все остальные окна в графическом интерфейсе называются немодальными.

4.4.7. Тексты и диалоги

Важной составляющей системы являются различные **надписи** в формах и окнах диалога. Некоторые принципы, которыми необходимо руководствоваться при создании текстовых диалогов и отображений, приведены ниже:

- текст в нижнем регистре читается приблизительно на 13 % быстрее, чем текст, напечатанный полностью в верхнем регистре;
- символы верхнего регистра наиболее эффективны для отображения информации, которая должна привлечь внимание. Не следует использовать ВЕРХНИЙ РЕГИСТР для выделения какой-либо информации;

• выровненный по правому краю текст труднее читать, чем равномерно распределенный текст с невыровненным правым полем;

• оптимальный интервал между строками должен быть равен или немного больше, чем высота символов.

4.4.8. Элементы управления

Элементы управления — общий термин для компонентов интерфейса типа кнопок, переключателей и т. д., которые служат пользователю для осуществления каких-либо действий в системе (ввода информации, вызова функций и т. д.).

Кнопки используются в системе для выбора какой либо опции или для вызова события (например, запуск подпрограммы).

Переключатели подобны кнопкам выбора, в которых пользователь выбирает значение из фиксированного списка, но в данном случае пользователь может выбрать более чем одно значение из списка.

Скролеры или полосы прокрутки могут быть помещены в горизонтальную или вертикальную линейку в форме, отчете или текстовом поле и служат для доступа к невидимой на экране в данный момент информации.

Метки и текстовые блоки используются для текстовой информации. Различие между ними следующее: текстовые поля, позволяют пользователю вводить текстовые данные в поля, в то время как метки являются нередактируемыми полями, используемыми только для отображения текста, типа подсказок, команд пользователя и т. д.

Списки — специализированные средства управления, которые отображают раскрывающиеся списки значений (часто с присоединенными скролерами, для перемещения по списку) и позволяют пользователю выбирать значение из списка или вводить другое значение в присоединенное текстовое поле. Списки — удобный и компактный элемент интерфейса, занимающий минимум места на экране и в то же время несущий большую информационную нагрузку.

4.4.9. Дизайн заголовков и полей

Ниже представлены рекомендации по созданию и размещению полей и заголовков в формах и отчетах:

- для отдельных полей заголовок должен быть выровнен по левому краю; числовых полей — по правому краю; для полей списков заголовок должен быть выше и левее по основного поля;
- длинные колоночные поля или длинные столбцы информационных единиц с одиночными полями необходимо объединять в группы по пять элементов, разделяемых пустой строкой. Это помогает пользователю мысленно обрабатывать информацию по выделенным группам;
- в формах с большим количеством информации необходимо использовать названия разделов, которые однозначно свидетельствуют о характере принадлежащей им информации;
- необходимо четко разделить отображение заголовков и непосредственно полей ввода, поскольку такая путаница может вызвать дискомфорт у пользователя;
- заголовки должны быть краткими, знакомыми пользователю и содержательными;
- поля, необязательные для заполнения либо не имеющие особой важности, должны отличаться визуально (цветом или другими эффектами) от полей важных и обязательных для заполнения.

4.4.10. Форматы ввода

Необходимо обеспечить ввод значений по умолчанию во все поля, которые это допускают. Можно назначить клавиши или коды для ввода часто повторяющихся значений. Входные данные должны быть значимыми и общепринятыми.

Не рекомендуется объединять поля ввода чисел и символов, поскольку числовые и алфавитные клавиши расположены относительно неудобно друг от друга на клавиатуре.

Следует минимизировать размер полей в формах, насколько это возможно.

Необходимо исключить частое переключение между верхним и нижним регистрами для ускорения ввода данных.

Не нужно требовать от пользователя ввода незначимых цифр (например, вместо 00000010 вводить только 10).

4.4.11. Организация системы навигации и системы отображения состояний

Навигация обеспечивает пользователю способность перемещаться между различными экранами, информационными единицами и подпрограммами в автоматизированной системе. В полноценной системе пользователь всегда может получить информацию о состоянии системы, процесса выполнения или активной подпрограмме.

Существует ряд навигационных средств и приемов, которые помогают пользователю ориентироваться в системе. Они включают: использование заголовков страниц для каждого экрана; использование номеров страниц, строк и столбцов; отображение текущего имени файла вверху экрана. Тип системы навигации зависит от принятого стиля интерфейса. В интерфейсах с меню можно использовать иерархически-структурированное меню. Для выхода из подменю нужно применять простые действия, учитывая, что диалоговые интерфейсы сами по себе защищают пользователя от ошибочных действий. Информация состояния обычно отображается внизу экрана и содержит в себе данные о количестве записей, числе обработанных единиц, процессе печати, очереди печати и т. д.

4.4.12. Проектирование сообщений

Сообщения необходимы для ориентирования пользователя в нужном направлении, для подсказок и предупреждений по выполнению необходимых действий на пути решения задачи. Они также включают подтверждения действий со стороны пользователя и подтверждения того, что задачи были выполнены системой успешно либо по каким-то причинам не выполнены. Сообщения могут быть представлены в форме диалога, экранных заставок и т. п. Сообщение об отказе (повреждении) оборудования и подсистем, предупреждение (прогноз) о возможных негативных последствиях называется ***аларм***. Ниже представлены рекомендации по формированию сообщений:

• каждое сообщение должно начинаться кратким и толковым, понятным пользователю описанием ситуации: что произошло, почему произошло. Как правило, далее должно следовать предложение о способе решения проблемы;

• сообщения должны быть консистентными с другими сообщениями программной системы по содержанию, структуре и стилю (тону);

• сообщения должны быть кратки и конкретны, привлекать внимание пользователя системы. Более полную информацию можно изложить в так называемых ***логах*** **(**дополнительных файлах системы) или дать ссылку на справочную систему. Для совместимости с малыми дисплеями принято использовать в одном сообщении не более 150 символов. Следует учитывать изменение длины сообщения при языковых локализациях;

• в сообщениях лучше использовать конструктивный и позитивный тон. Ободрение и обнадёживание при этом приветствуются. Неуместны порицание, неодобрение, осуждение, упрек, обвинение, возложение ответственности за произошедшее на пользователя, представление действий пользователя в качестве причины неполадок в работе системы;

• не следует использовать двойное отрицание при построении сообщений, таких как «Нет компонент, которые не используются»;

• описывать текущую ситуацию в сообщениях лучше настоящим временем;

• необходимо использовать терминологию, знакомую пользователям;

Сообщения могут предложить пользователю:

• выбрать из предложенных альтернатив некую опцию или набор опций;

• ввести некоторую информацию;

• выбрать опцию из набора опций, которые могут изменяться в зависимости от текущего контекста;

• подтвердить фрагмент введенной информации перед продолжением ввода.

Сообщения могут быть помещены в модальные диалоговые окна, которые вынуждают пользователя ответить на вопрос

прежде, чем может быть предпринято любое другое действие, потому что все другие средства управления заморожены. Это может быть полезно, когда система должна вынудить пользователя принять решение перед продолжением работы. Немодальные диалоговые окна позволяют работать с другими элементами интерфейса, в то время как само окно может игнорироваться.

4.4.13. Таблицы

Табличное представление информации, как правило, предназначено не для последовательного чтения, а в основном для облегчения выборочного поиска нужных сведений. Желательна горизонтальная организация таблиц. Категорически недопустимо смешанное в рамках одного документа, пусть даже и на разных страницах, вертикальное и горизонтальное расположение считываемой информации. По ширине таблицы не должны растягиваться более чем на одну страницу. Буквенно-цифровые данные в них следует выравнивать слева, а числовые справа, до десятичной запятой.

К текстовой информации в таблицах предъявляются свои специфические требования:

- использование простых, односложных, неопределенно-личных или безличных предложений, не имеющих эмоциональной окраски;
- порядок слов в предложении – прямой;
- отсутствие сложных предложений с длинным рядом последовательных подчинений;
- рекомендуемая длина сообщений 7–11 значащих слов;
- минимальное количество логических связок И/ИЛИ и их сочетаний;
- отсутствие многоступенчатых подчиненных предложений с неоднократно повторяемыми частицами НЕ (чем часто грешат дословно переведенные тексты);
- отсутствие аббревиации в ключевых словах, сокращение которых может привести к искажению смысла.

4.5. Эргономическая экспертиза

Эргономическая экспертиза — комплекс научно-технических и организационно-методических мероприятий по оценке выполнения в проектных, предпроектных и рабочих документах и в образцах эргономических требований технического задания, нормативно-технических документов, а также по разработке рекомендаций для устранения отступлений от этих требований.

Такая экспертиза проводится при обосновании выполнения каждого этапа опытно-конструкторской разработки: технического предложения, эскизного проекта, технического проекта, рабочего проекта. Материалы экспертизы — акт либо протокол — включаются в документы, представляемые сдаче системы в промышленную эксплуатацию.

Серьёзная эргономическая экспертиза программного продукта (usability testing) — сложное и дорогостоящее мероприятие, проводимое по специальным методикам и позволяющее получить как качественные, так и количественные оценки эргономичности программного продукта в целом его важных компонент (пользовательского интерфейса и пользовательской документации).

Предварительная эргономическая экспертиза заметно снижает вероятность последующих переделок продукта. Обычно более восьмидесяти процентов изменений в проекте приходится именно на интерфейс. Кроме того, профессиональная разработка пользовательского интерфейса снижает стоимость последующей технической поддержки и дает определенную гарантию от проявлений недовольства пользователей системой.

Контрольные вопросы

1. Перечислите основные эргономические проблемы, возникающие при разработке АСОИУ.
2. Перечислите основные принципы проектирования эргономичного интерфейса.
3. Опишите основные правила формирования сообщений, возникающих в системе.

5. ДОКУМЕНТИРОВАНИЕ АСОИУ

5.1. Общие сведения о документации АСОИУ

В процессе создания системы разрабатывается пакет документов, включающий набор материалов по обеспечению ЖЦ создания ИС, начиная с технических предложений по созданию системы и заканчивая руководством по установке и настройке системы. В настоящее время разработаны и применяются стандарты на ведение различных типов документации. Существует более шестидесяти типовых шаблонов документов. Документирование АСОИУ является важным аспектом при создании информационных систем и должно выполняться в соответствии с действующими стандартами и нормативными документами. Полная и всеобъемлющая документация на систему способствует повышению качества АСОИУ в целом.

Требования к содержанию документов, разрабатываемых при создании АСОИУ, установлены РД 50-34.698-90 (редакция от 14 февраля 2005 года), а также соответствующими государственными стандартами Единой системы программной документации (ЕСПД), Единой системы конструкторской документации (ЕСКД), Системы проектной документации для строительства (СПДС) и ГОСТ 34.602.

Содержание документов является общим для всех видов АИС и при необходимости может дополняться разработчиком документов в зависимости от особенностей создаваемой АИС. Допускается включение в документы дополнительных разделов и сведений, объединение и исключение разделов.

Содержание каждого документа, разрабатываемого при проектировании АИС согласно ГОСТ 34.201, определяет разработчик в зависимости от объекта проектирования (системы, подсистема и т. д.).

Содержание документов, разрабатываемых на предпроектных стадиях по ГОСТ 34.601, и организационно-распорядительных документов определяют разработчики в зависимости от объема информации, необходимой и достаточной для дальнейшего использования документов.

Документы при необходимости сброшюровывают в книги или тома, к которым составляют описи.

В этом разделе рассмотрим состав и назначение основной документации, создаваемой на всех этапах жизненного цикла АСОИУ, наличие которой способствует повышению качества создаваемых систем. Поскольку документы, формируемые в ходе создания системы, отражают сущность процессов, протекающих на конкретном этапе жизненного цикла системы, важным фактором здесь является качество и полнота разрабатываемой документации.

По своему назначению можно выделить следующие типы документации:

проектная и общесистемная документация. Эта документация разрабатывается менеджером проекта, аналитиком и менеджером по маркетингу. В ряде случаев для составления проектной документации могут быть задействованы ведущие программисты. На основе этой документации определяется принцип создания системы;

техническая документация, созданием которой занимаются технические писатели, в функции которых входит также создание справочной документации, являющейся составной программной частью системы. В том случае если разрабатывается узкоспециализированная документация, не предназначенная заказчику, к созданию технической документации могут быть также привлечены члены группы разработки программного обеспечения.

Техническую документацию, в свою очередь, можно разделить на ***пользовательскую*** и ***технологическую*** (***внутреннюю***) ***документацию***. Пользовательская документация предназначена для различных уровней пользователей и необходима для эксплуатации системы заказчиком. Такую документацию называют также ***эксплуатационной***. Технологическая документация необходима для контроля реализации проекта, а также может быть использована для дальнейшего развития системы и предназначена для специалистов занятых в разработке системы;

финансово-организационная документация, за создание которой помимо бухгалтерских служб отвечают менеджер про-

екта, менеджер по маркетингу и руководитель компании. В состав этого типа документации может быть включен, например, план реализации разработки, в котором выделяются основные цели создания будущей системы.

5.2. Проектная и общесистемная документация

5.2.1. Технические предложения

Технические предложения на создание информационной системы разрабатываются на стадии планирования системы по завершении этапа обследования объекта автоматизации. В этом документе отражается концептуальное видение проблемы автоматизации, определяются цели работы, описываются общие требования к ожидаемым результатам.

Кроме этого, в технических предложениях определяются базовые требования к функциям, выполняемым системой, требования к программно-аппаратному обеспечению и составу технической документации. В завершение приводится состав и содержание работ по созданию системы.

5.2.2. Техническое задание

Техническое задание (ТЗ) на создание АСОИУ разрабатывает организация-разработчик системы на основании технических требований (заявки заказчика, технических предложений и т. п.) в соответствии с ГОСТ 34.602-89 (см. прил. 1) при непосредственном участии заказчика. **Техническое задание** является основным документом, в соответствии с которым проводят создание АСОИУ и приемку его заказчиком. Основанием для технического задания, кроме технических предложений, служит технико-экономическое обоснование, в котором должны найти отражение все положения, рассмотренные на этапе концептуального проектирования (полное описание содержания формы данного документа см. ГОСТ 11.091.655-84).

Техническое задание содержит следующие разделы:

- характеристика объекта из существующей системы управления;

- цели, функции, задачи создания автоматизированной системы;
- ожидаемые технико-экономические результаты;
- выводы и предложения.

«Введение» должно содержать: предполагаемые сроки; периоды, содержащие работы; объемы, порядок финансирования и источники; ссылки на методически нормированные документы.

Раздел «Характеристика объекта» включает в себя общие характеристики объекта: структура системы управления; описание функций объекта и его структур, функций управления; используемые методы управления; действующая система документооборота; перечень выявленных недостатков и способы их устранения; готовность объекта к внедрению.

В разделе «Экономическая эффективность» указывается перечень основных источников экономической эффективности и ожидаемые затраты.

В разделе «Выводы и предложения» находится сопоставление ожидаемых результатов внедрения автоматизированной системы с заданными целями и критериями ее развития. Выводы о целесообразности создания системы и описание концептуальной модели.

При конкурсной организации работ варианты проекта ТЗ на создание АСОИУ рассматриваются заказчиком, который либо выбирает предпочтительный вариант, либо на основании сопоставительного анализа подготавливает с участием будущего разработчика АСОИУ окончательный вариант ТЗ на создание системы.

Работу по согласованию проекта ТЗ осуществляют совместно разработчик ТЗ и заказчик системы, каждый в организациях своего министерства (ведомства).

Замечания по проекту ТЗ должны быть представлены с техническим обоснованием. Решения по замечаниям должны быть приняты разработчиком проекта ТЗ и заказчиком системы до утверждения ТЗ.

Если при согласовании проекта ТЗ возникли разногласия между разработчиком и заказчиком (или другими заинтересованными организациями), то составляется протокол разногласий

в произвольной форме и конкретное решение принимается в установленном порядке.

Утверждение ТЗ осуществляют руководители предприятий (организаций) разработчика и заказчика системы. ТЗ до передачи его на утверждение должно быть проверено службой нормоконтроля организации-разработчика ТЗ и при необходимости подвергнуто метрологической экспертизе.

Копии утвержденного ТЗ в 10-дневный срок после утверждения высылаются разработчиком ТЗ участникам создания системы. Изменения к ТЗ не допускается утверждать после представления созданной системы или ее очереди на приемосдаточные испытания для принятия ее в опытную или промышленную эксплуатацию.

Качество будущей системы напрямую зависит от полноты проработки технического задания, поэтому на разработку ТЗ отводится довольно большой промежуток времени в общем периоде разработки системы.

5.2.3. Исходная спецификация на систему

Спецификация (технические требования) — это документ, который детально излагает полные, точные, поддающиеся проверке требования, стратегию проектирования или другие характеристики элементов системы или системы в целом, а также методики проверки выполнения условий спецификации. Существуют различные разновидности спецификаций, например, спецификация требований к информационной системе или спецификация разработки.

Спецификация системы должна давать полное представление о функциях, выполняемых программным обеспечением. Этот документ должен быть подготовлен до начала реализации этапа проектирования и программирования.

Документ должен быть полным, последовательным, достаточно подробным и выполненным на уровне современных требований к АСОИУ и, по возможности, не содержать излишних подробностей. Требования должны быть недвусмысленными, поддающимися легко реализуемым испытаниям или проверке.

В документе, содержащем требования к системе, должно быть проведено четкое разграничение между существенными требованиями и менее жесткими (второстепенными) требованиями.

Применение формального языка спецификаций может обеспечить наглядность представления и полноту функциональных требований к системе.

Функциональные требования к системе, изложенные в спецификации, содержат перечень функций, которые должны быть реализованы системой. При этом необходимо определить функции, которые должны быть выполнены, а не средства их реализации. Должно быть обеспечено подробное описание всех функций системы с указанием их связей друг с другом, а также с выходными и входными параметрами системы. Должны быть разработаны диаграммы, отражающие функциональные связи, а также входные/выходные зависимости.

В описании функций содержится:

- обоснование выбора определенных функций;
- условия, вызывающие исполнение каждой функции;
- последовательность задач, действий или событий;
- начальные условия, состояние системы при инициализации функции;
- возможности дальнейшего расширения функции;
- подробности процедуры верификации.

В спецификации указываются следующие характеристики системы:

- наиболее благоприятные/неблагоприятные режимы функционирования системы;
- планируемый показатель качества функционирования, включая точность;
- временные характеристики;
- другие существующие ограничения и обязательные условия;
- функции обработки ввода/вывода, протокол синхронизации передачи данных;

• функции подтверждения правильности ввода (например, подтверждения правильности формата, поля, проверки логики и подтверждения правильности источника).

5.2.4. Проектная оценка надежности системы

Согласно РД 50-34.698-90 проектная оценка надежности автоматизированной информационной системы содержит следующие разделы:

• введение;
• исходные данные;
• методику расчета;
• расчет показателей надежности;
• анализ результатов расчета.

Во «Введении» указывают:

• назначение расчета надежности системы;
• перечень оцениваемых показателей надежности;
• состав учитываемых при расчете факторов, а также принятые допущения и ограничения.

В разделе «Исходные данные» приводят:

• данные о надежности (паспортные и справочные) элементов АИС, учитываемые при расчете надежности системы;
• данные о режимах и условиях функционирования элементов АИС;
• сведения об организационных формах, режимах и параметрах эксплуатации АИС.

В разделе «Методика расчета» приводится обоснование выбора методики расчета и нормативно-технический документ, согласно которого проводят расчет, или краткое описание методики расчета и ссылку на источники, где она опубликована.

В разделе «Расчет показателей надежности» указывают:

• надежностные структуры компонентов АИС (комплекса технических средств, программного обеспечения и персонала) по всем оцениваемым функциям (функциональным подсистемам) АИС;
• необходимые вычисления;
• результаты расчета.

В разделе «Анализ результатов расчета» указывают:

• итоговые данные расчета по каждой оцениваемой функции (функциональной подсистеме) АИС и каждому нормируемому показателю надежности;

• выводы о достаточности или недостаточности полученного уровня надежности АИС по каждой оцениваемой функции (функциональной подсистеме) АИС и, при необходимости, рекомендации по повышению надежности.

Если при оценке надежности АИС нельзя учесть уровень надежности программного обеспечения АИС и уровень надежности действий персонала АИС, то в документе «Проектная оценка надежности системы» указывают сведения по оценке надежности АИС только с учетом надежности комплекса технических средств, в том числе нестандартных.

5.2.5. Программа и методика испытаний АСОИУ

Программа и методика испытаний системы в целом или ее подсистем согласно РД 50-34.698-90 на этапе опытной эксплуатации предназначена для установления данных, обеспечивающих получение и проверку проектных решений, выявление причин сбоев, определение качества работ, показателей качества функционирования системы, проверку соответствия системы требованиям техники безопасности, продолжительность и режим испытаний.

Этот документ должен содержать перечни конкретных проверок, которые следует осуществлять при испытаниях для подтверждения выполнения требований ТЗ, со ссылками на соответствующие методики испытаний.

В программу испытаний включается перечень проверок:

• соответствие системы ТЗ;

• комплектность системы;

• комплектность и качество документации;

• комплектность, достаточность состава программных средств и программной документации и уровень их качества;

• количество и квалификация обслуживающего персонала;

• степень выполнения требований функционального назначения системы;

• контролепригодность системы;

• выполнение требований техники безопасности, противопожарной безопасности, промышленной санитарии, эргономики;

• функционирование системы с применением программных средств.

Описание методов испытаний системы по отдельным показателям рекомендуется располагать в той же последовательности, в которой они расположены в технических требованиях.

Программа испытаний содержит разделы:

• объект испытаний;

• цель испытаний;

• общие положения;

• объем испытаний;

• условия и порядок проведения испытаний;

• материально-техническое обеспечение испытаний;

• метрологическое обеспечение испытаний;

• отчетность.

В разделе «Объект испытаний» указывают:

• полное наименование системы, обозначение;

• комплектность испытательной системы.

В разделе «Цель испытаний» указывают конкретные цели и задачи, которые должны быть достигнуты и решены в процессе испытаний.

В разделе «Общие положения» указывают:

• перечень руководящих документов, на основании которых проводят испытания;

• место и продолжительность испытаний;

• организации, участвующие в испытаниях;

• перечень ранее проведенных испытаний;

• перечень предъявляемых на испытания документов, откорректированных по результатам ранее проведенных испытаний.

В разделе «Объем испытаний» указывают:

• перечень этапов испытаний и проверок, а также количественные и качественные характеристики, подлежащие оценке;

• режим испытаний и последовательность их проведения;

• требования по испытаниям программных средств;

• перечень работ, проводимых после завершения испытаний, требования к ним, объем и порядок проведения.

В разделе «Условия и порядок проведения испытаний» указывают:

- условия проведения испытаний;
- условия начала и завершения отдельных этапов испытаний;
- имеющиеся ограничения в условиях проведения испытаний;
- требования к техническому обслуживанию системы;
- меры, обеспечивающие безопасность и безаварийность проведения испытаний;
- порядок взаимодействия организаций, участвующих в испытаниях;
- порядок привлечения экспертов для исследования возможных повреждений в процессе проведения испытаний;
- требования к персоналу, проводящему испытания, и порядок его допуска к испытаниям.

В разделе «Материально-техническое обеспечение испытаний» указывают конкретные виды материально-технического обеспечения с распределением задач и обязанностей организации, участвующих в испытаниях.

В разделе «Метрологическое обеспечение испытаний» приводят перечень мероприятий по метрологическому обеспечению испытаний с распределением задач и ответственности организаций, участвующих в испытаниях.

В разделе «Отчетность» указывают перечень отчетных документов, которые должны оформляться в процессе испытаний и по их завершению, с указанием организаций и предприятий, их разрабатывающих, согласующих и утверждающих, и сроки оформления этих документов. Отчетными документами являются акт и отчет о результатах испытаний, акт технического состояния системы после испытаний.

При проведении испытаний в несколько этапов программы испытаний должны быть оформлены в виде единого документа. Методики испытаний разрабатывают на основе ТЗ и утвержденных программ испытаний с использованием типовых методик испытаний. При этом отдельные положения типовых методик испытаний могут уточняться и конкретизироваться в разрабатываемых методиках испытаний в зависимости от особенности

системы и условий проведения испытаний. Содержание разделов методик устанавливает разработчик.

5.3. Пользовательская документация

5.3.1. Состав пользовательской документации

В соответствии с [17] типичным является следующий состав пользовательской документации для АСОИУ:

общее описание системы, содержащее краткую характеристику функциональных возможностей разработанной системы. На основе этого документа пользователь должен получить общее представление о системе;

руководство по управлению системой, предназначенное для администраторов АСОИУ;

руководство пользователя (инструкция по применению), предназначенное для конечных пользователей и содержащее необходимую информацию по применению системы, организованную в форме удобной для ее изучения. Если происходит развитие системы, то при разработке пользовательской документации необходимо учитывать все изменения, внесенные в систему с момента последней выдачи комплекта документов пользователю и оперативно вносить соответствующие изменения в текст документации.

Рассмотрим более подробно назначение пользовательской документации, разрабатываемой в ходе создания системы.

5.3.2. Общее описание системы

Общее описание системы предоставляется заказчику при передаче системы в эксплуатацию и содержит в себе общее функциональное описание системы (описание основных реализуемых функций), назначение системы и область ее применения. Кроме этого в общее описание системы может быть включен перечень программно-аппаратных требований к функционированию системы.

Если система тиражируется и предполагается ее дальнейшее продвижение на рынок, то в состав этого документа может быть включен перечень организаций, в которых функционирует

представляемая система. Также в этом документе могут содержаться сведения о разработчиках и координаты исполнителей.

5.3.3. Руководство по управлению системой

Руководство по управлению системой предназначено для системных администраторов. В документе содержатся требования к программно-аппаратной среде, сведения по инсталляции системы и ее настройке на рабочих местах пользователей.

Кроме того, данный документ может содержать описание сообщений, генерируемых при взаимодействии АСОИУ с другими системами, и рекомендации о том, как реагировать на эти сообщения. Если программное средство использует специальную системную аппаратуру, документ должен содержать информацию о ее настройке и сопровождении.

5.3.4. Руководство пользователя

Руководство пользователя должно представлять собой подробную инструкцию по работе с системой. Разработка пользовательской документации начинается сразу после создания внешнего описания. Качество этой документации может существенно определять успех программы. Она должна быть достаточно проста и удобна для пользователя. Хотя черновые варианты пользовательских документов создаются основными разработчиками программного средства, к созданию их окончательных вариантов необходимо привлекать профессиональных технических писателей.

Удачный пользовательский документ существенно зависит от точного определения аудитории, для которой он предназначен. Пользовательская документация должна содержать информацию, необходимую для каждой аудитории.

Здесь можно использовать такое понятие как режим использования. Под ***режимом использования документа*** понимается способ, определяющий, каким образом используется этот документ. Обычно пользователю достаточно больших программных систем требуются либо документы для изучения системы (*инструкции*), либо для уточнения некоторой информации (*справочники*) [17].

Для обеспечения качества пользовательской документации разработан ряд стандартов, в которых предписывается порядок разработки этой документации, формулируются требования к каждому виду пользовательских документов и определяются их структура и примерное содержание.

5.4. Внутренняя документация системы

5.4.1. Спецификация тестирования системы

Спецификация тестирования системы является одним из главных документом плана верификации этапа программирования. Этот документ должен базироваться на результатах детального изучения функциональных требований к системе и содержать подробную информацию по тестированию каждого из ее компонентов. Спецификация тестирования системы основана на общем описании тестируемой АСОИУ. Описание составляется группой проектирования. Спецификация тестирования системы должна включать:

- окружение, в котором выполняется тестирование;
- процедуры тестирования;
- критерии приемки, т.е. детальное изложение критериев, которым должны удовлетворять при приемке модули и основные компоненты АСОИУ на уровне подсистем;
- процедуры обнаружения ошибок и корректирующие действия;
- список необходимой документации, которая должна быть создана при верификации этапа программирования.

5.4.2. Отчет о тестировании системы

В отчете о тестировании системы приводятся результаты выполнения тестирования в соответствии со спецификацией тестирования. В нем дается заключение о том, в какой мере система удовлетворяет требованиям к качеству функционирования, указанным в функциональных требованиях к ней. Данный документ содержит перечень результатов этапов тестирования. Он должен также включать заключения обо всех недостатках системы и расхождениях, выявленных в процессе ее тестирования.

Отчет о тестировании системы должен содержать следующие пункты, касающиеся модулей и более высоких уровней проекта:

- конфигурация системы, используемая при тестировании аппаратуры, если в состав АСОИУ входит аппаратное обеспечение;
- список тестируемых входных данных;
- список тестируемых выходных данных;
- дополнительные данные (синхронизация, последовательность событий и др.);
- соответствие критериям приемки, указанным в спецификации тестирования;
- протокол регистрации выявленных ошибок, в котором приводится описание типа ошибки и возможные способы ее устранения, используемые группой проектирования и разработки.

5.4.3. Руководство программиста

Руководство программиста или документация по сопровождению программного средства описывает программное средство с точки зрения ее разработки. Эта документация необходима в том случае, если предполагается дальнейшая модернизация системы. При этом следует учесть, что сопровождение и модернизация системы может производиться другими программистами, при необходимости к модернизации системы привлекается специальная команда разработчиков-сопроводителей. Этой команде придется иметь дело с такой же документацией, которая определяла деятельность команды первоначальных (основных) разработчиков системы, — с той лишь разницей, что эта документация для команды разработчиков-сопроводителей будет, как правило, чужой (она создавалась другой командой).

Команда разработчиков-сопроводителей должна изучить эту документацию, чтобы понять строение и процесс разработки модернизируемого программного средства и внести в эту документацию необходимые изменения, повторяя в значительной степени технологические процессы, с помощью которых создавалась полученная ими версия системы.

Документацию по сопровождению системы можно разбить на две группы [17]:

1) документация, определяющая строение программ и структур данных АСОИУ и технологию их разработки;

2) документация, помогающая вносить изменения в систему.

Документация первой группы содержит итоговые документы каждого технологического этапа разработки системы. Она включает следующие документы:

1) внешнее описание системы;

2) описание архитектуры системы, включая внешнюю спецификацию каждой ее программы;

3) описание модульной структуры для каждой программы системы, включая внешнюю спецификацию каждого включенного в нее модуля;

4) спецификация и описание строения для каждого модуля;

5) тексты модулей на выбранном языке программирования;

6) документы установления достоверности системы, описывающие процесс установления достоверности каждой программы АСОИУ. Документы установления достоверности включают, прежде всего, документацию по тестированию (схема тестирования и описание комплекта тестов), но могут включать и результаты других видов проверки программного средства, например доказательства свойств программ.

Документация второй группы содержит рекомендации по дальнейшему сопровождению системы. Общая проблема сопровождения программного средства — обеспечение согласованности всех изменений. Для ее решения связи и зависимости между документами и их частями должны быть зафиксированы в базе данных управления конфигурацией.

Если при сдаче готовой системы заказчику передается исходный код программ, то в состав пользовательской документации при сдаче системы в промышленную эксплуатацию включается руководство программиста.

5.4.4. Описание структуры и глоссарий базы данных

Описание структуры и глоссарий базы данных следует создавать с начального этапа проектирования базы данных системы, т. е. с момента разработки концептуальной модели предметной области. В этом документе должны быть представлены

все уровни представлений БД, описаны все сущности и связи предметной области.

Описание структуры и глоссарий базы данных необходимы для ведения полноценной эволюции схемы базы данных и предназначены для администратора базы данных и разработчиков, занятых реализацией той части системы, которая взаимодействует с базой данных.

5.5. Дополнительная документация

Следует отметить, что набор документов на систему не ограничивается перечисленными выше документами. Кроме представленного в данном разделе перечня документов в процессе разработки системы могут формироваться различные сопроводительные и финансовые документы:

- договора на создание системы;
- протоколы разногласий;
- протоколы проведения совещаний разработчиков;
- аннотированный отчет о проведенных работах по созданию системы;
- акт завершения работ;
- акт приемки в опытную эксплуатацию;
- акт приемки в промышленную эксплуатацию;
- протокол испытаний и др.

Также в состав документации на систему может быть включена документация, определенная конкретными потребностями и условиями разработки. В любом случае между заказчиком и исполнителем оформляется соглашение о составе, структуре и содержании документации, разрабатываемой при создании системы.

5.6. Стандартизация качества служебной информации

Одним из вопросов качества АСОИУ является вопрос качества служебной информации, появляющейся в результате межсистемного и внутрисистемного взаимодействия. Вопросам оценки и обеспечения качества служебной информации посвящена серия из государственных стандартов России и рекомендаций по стандартизации (ГОСТ Р 51167–98, ГОСТ Р 51171-98, ГОСТ Р

50.1.15–98, ГОСТ Р 50.1.017–98). К сожалению, эта серия нормативных документов в справочниках не выделена в отдельный раздел, а разбросана поодиночке по различным разделам. Поэтому целесообразно привести краткие сведения об этих стандартах и рекомендация [18].

ГОСТ Р 51170–98 «Качество служебной информации. Термины и определения» посвящен основным понятиям в области качества служебной информации. Приведены не только свойства служебной информации, определяющие ее качество, но и соответствующие количественные показатели качества данных.

ГОСТ Р 51168–98 «Качество служебной информации. Условные обозначения элементов технологических процессов переработки данных» регламентирует условные обозначения элементов технологических процессов переработки данных в задачах оценки и обеспечения безошибочности и временных свойств служебной информации. Приведены наименования, обозначения, определения обязательных символов основных технологических операций и содержание буквенных обозначений. Указаны особенности применения условных обозначений технологических операций в задачах оценки и обеспечения качества служебной информации.

ГОСТ Р 51167–98 «Качество служебной информации. Графические модели технологических процессов переработки данных» устанавливает наиболее часто употребляемые графические модели технологических процессов переработки данных (ТППД) в задачах оценки и обеспечения безошибочности и временных свойств служебной информации, а также в задачах планирования и контроля за ходом выполнения ТППД. Термины, применяемые в настоящем стандарте, — по ГОСТ Р 51170. Условные обозначения элементов ТППД в задачах обеспечения безошибочности и временных свойств служебной информации — по ГОСТ Р 51168.

В стандарте рассмотрены особенности типовых моделей ТППД в виде сетевых графиков, логико-сетевых графов, информационных цепей, ленточных и линейных графиков, операграмм, схем алгоритмов, программ, данных и систем, а также сетей Петри.

ГОСТ Р 51169–98 «Качество служебной информации. Система сертификации информационных технологий в области качества служебной информации. Термины и определения» устанавливает термины и определения основных понятий по сертификации служебной (технологической и официальной) информации,

Под сертификацией информационной технологии в области качества служебной информации понимается действие третьей стороны, доказывающее, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная информационная технология соответствует конкретному стандарту или другому нормативному документу в области качества служебной информации.

В приложении приведены основные этапы сертификации информационных технологий в области качества служебной информации. Дана блок-схема алгоритма процедуры аттестации информационной технологии в области качества служебной информации.

ГОСТ Р 51171–98 «Качество служебной информации. Правила предъявления информационных технологий на сертификацию» распространяется на информационные технологии всех видов служебной информации, используемой в государственной, производственной и коммерческой деятельности.

Стандарт устанавливает основные правила предъявления информационных технологий на обязательную сертификацию в области качества служебной информации. Стандарт рекомендуется применять и при добровольной сертификации информационных технологий в области качества служебной информации.

Приведены общие правила сертификации, а также правила предъявления информационных технологий на аттестацию и на сертификацию систем управления качеством служебной информации. Представлены форма декларации-заявки на проведение сертификации информационных технологий, анкета-вопросник (сведения о состоянии информационной технологии).

Р 50.1.015–98 «Качество служебной информации. Методика оценки безошибочности по технологическим схемам переработки информации» содержит рекомендации по оценке безоши-

бочности служебной информации и распространяется на технологические процессы переработки служебной информации, в которых можно выделить отдельные последовательно выполняемые технологические операции.

Наряду с общими положениями рассмотрены особенности составления и разметки информационной цепи, последовательность действий при оценке безошибочности данных и анализе результатов оценки. Приведен пример оценки безошибочности данных.

ГОСТ Р 50.1.016–98 «Качество служебной информации. Графические модели в задачах выявления и анализа факторов, влияющих на технологические процессы переработки служебной информации» содержит наиболее распространенные графические модели: схемы причинно-следственных связей, диаграммы видов нарушений технологического процесса, деревья опасных событий, графические модели при расчетах надежности технических объектов, графические модели процессов возникновения отказов, модели совпадения времени действия дестабилизирующих факторов.

Рекомендации предназначены для работников систем сбора и переработки служебной информации и распространяются на все виды технологических процессов в этих системах.

ГОСТ Р 50.1.017–98 «Качество служебной информации. Методика оценки временных свойств по технологическим схемам переработки информации» содержит рекомендации по оценке оперативности и идентичности служебной информации. Методика распространяется на технологические процессы переработки данных, в которых можно выделить отдельные последовательно или параллельно выполняемые технологические операции.

Наряду с общими положениями определен алгоритм выбора метода оценки оперативности данных и основные особенности различных методов оценки оперативности: сетевого планирования и управления, вариантов метода оценки надежности систем работ и др. Описаны основные особенности методов оценки идентичности данных: расчетного метода и моделирования.

В указанной серии стандартов и Рекомендаций по стандартизации рассмотрены в основном технические составляющие качества служебной информации.

Контрольные вопросы

1. Приведите классификацию документации на систему.
2. Поясните назначение проектной документации.
3. Что содержится в пользовательской документации?
4. Что содержится во внутренней документации системы?

6. ТЕСТИРОВАНИЕ АСОИУ

6.1. Верификация и валидация системы

На каждом этапе разработки система подвергается тестированию. В этом разделе рассматриваются предпосылки и методы проведения тестирования. Следует учесть, что недостаточно полно протестированная система с большой долей вероятности будет плохо функционировать, поэтому тестированию систем во всех компаниях, разрабатывающих АСОИУ, уделяется особое внимание. Тестирование и оценка работоспособности системы, которые обеспечивают подтверждение ее соответствия требованиям на функциональные и эксплуатационные свойства, а также соответствия требованиям к интерфейсу, называется **валидацией системы**. Одной из составляющих частей тестирование системы является ее верификация на всех этапах разработки. **Верификация** — это процесс определения соответствия системы на каждом этапе разработки требованиям, устанавливаемым на предыдущих этапах.

Верификация, которая затрагивает вопросы соответствия функциональных требований системы (см. п. 1.2.2), выполняется после формулирования функциональных требований к системе и перед началом следующего этапа,

После завершения этапа проектирования и перед переходом к следующему этапу выполняется верификация, целью которой является проверка соответствия разработанного программного обеспечения системы спецификациям качества функционирования и функциональным требованиям к системе.

После завершения этапа программирования и перед переходом к следующему этапу выполняется верификация, целью которой является проверка степени проработанности создаваемого программного обеспечения системы и соответствия спецификации качества функционирования системы, составленной на этапе проектирования.

Таким образом, каждая фаза разработки системы должна завершаться процедурой верификации.

На этапе проектирования, как и на этапе планирования, программного кода еще нет, поэтому и здесь тестируются толь-

ко идеи. Однако на этом этапе идеи лучше формализованы и описаны намного подробнее, чем в первоначальных планах. Анализируя проектные документы, специалисты должны составить четкое представление о работе будущей системы.

Специалисты по тестированию могут и не участвовать в работе группы аналитиков, однако для планирования системы будущих тестов такое участие может быть необходимо. На совещаниях группы аналитиков специалистам по тестированию лучше всего быть пассивными участниками и высказываться только в случае необходимости. Одной из самых важных предпосылок успешного тестирования является правильная постановка задачи.

На этапе проектирования в центре внимания аналитиков должны быть следующие вопросы:

действительно ли проект хорош? Необходимо проанализировать возможность создания эффективного, компактного, хорошо тестируемого и легкого в сопровождении и модернизации продукта;

соответствует ли проект требованиям? Проект должен быть формализованным выражением требований, представленных в документации этапа планирования;

полон ли проект? Следует выявить наличие в проекте описаний всех взаимосвязей между модулями и данными, условий передачи данных между модулями и условий работы каждого модуля и их реализации;

достаточно ли он реалистичен? Необходимо рассмотреть степень удовлетворенности имеющихся системных ресурсов (как аппаратных, так и программных) потребностям проекта, возможность достаточно быстрой работы программного продукта, а также быстрого извлечения информации из баз данных и ее обработки, выяснить, насколько удачно выбраны инструментальные средства разработчиков;

хорошо ли описана в проекте подсистема обработки ошибок? Одной из ошибок разработчиков при нисходящем проектировании является желание оставить вопросы обработки ошибок, как самые незначительные, на завершающую стадию разработки системы. Однако о подобных элементах часто вооб-

ще забывают или же из-за нехватки времени они проектируются наспех, поверхностно и в результате плохо. Все возможные условия возникновения ошибок должны быть самым тщательным образом продуманы и описаны в проекте. Важно правильно определить уровень, на котором обрабатывается каждая из ошибок, чтобы ее возникновение в одном модуле не вело к ошибкам в других.

В ходе тестирования на этапе проектирования могут быть проведены следующие совещания:

совещания аналитиков. Обычно целью совещаний, проводимых при анализе проектных документов, является не решение проблем, а прежде всего их выявление. В таком совещании должна участвовать небольшая группа сотрудников — около семи человек. В эту группу не должны входить авторы проекта. Аналитики заранее читают документы и на совещании критикуют их и задают друг другу вопросы.

Во многих компаниях информационная система вообще не считается завершенной, пока на нее не будет составлена формальная рецензия (разумеется, одобрительная). Таким образом, проект перерабатывается и снова анализируется до тех пор, пока он не будет одобрен группой аналитиков. Совещания этой группы могут быть трех типов; обзорные, инспекционные и рецензионные;

обзорное совещание, на котором проектировщики демонстрируют модель программы. Шаг за шагом они показывают, что делает программа с тестовыми данными, предложенными аналитиками. Такая демонстрация позволяет увидеть, как взаимодействуют между собой различные части системы, и выявить ее недостатки (неудобные режимы, избыточность функций или пропущенные детали);

инспекционное совещание, на котором специалисты подробно анализируют каждый элемент проекта или его отдельный аспект (обработку ошибок, соответствие ранее выработанным стандартам, эффективность реализации конкретной функции и т. д.);

рецензионное совещание. К этому совещанию аналитики готовят список возникших у них вопросов. Они делятся своими соображениями и выделяют элементы проекта, которые показа-

лись им неточными или сомнительными. Цель этого совещания — сформировать список всех выявленных проблем и убедиться, что каждую из них проектировщики правильно поняли. Решение выявленных проблем в задачи совещания не входит.

Идеальное рецензионное совещание должно направляться опытным в этом деле специалистом и обязательно документироваться. Такой специалист-организатор находит подходящее помещение, ведет совещание, останавливает оппонентов, когда они прерывают друг друга или отклоняются от темы, и готовит итоговый отчет. Он следит за тем, чтобы от обсуждения проблем аналитики не переходили к обсуждению способов их решения. Это будет сделано позднее меньшей группой специалистов и вне рецензионного совещания.

Специальный персонал записывает все важные замечания и с помощью проекторов или другой аналогичной техники выводит их на большой экран, где они видны каждому участнику совещания. Любой, кому покажется, что записывающий упустил нечто важное, может попросить отобразить эту информацию. Обязательно должно фиксироваться каждое достигнутое соглашение. Записаны должны быть и все вопросы, которые остались открытыми до следующего совещания. Такая техника проведения совещаний способствует их продуктивности, особенно когда мнения аналитиков очень сильно расходятся.

Некоторые группы тестирования специально обучают свой персонал ведению и протоколированию таких совещаний. Качественное ведение протокола, способность выявить и зафиксировать основные стороны дискуссии является важным фактором, влияющим на дальнейшее обсуждение выявленных проблем.

6.2. Тестирование на стадии кодирования

На этапе кодирования программист пишет программы и сам их тестирует. Технология тестирования на этом этапе называется **тестированием «стеклянного ящика»** (*glass box*). Иногда ее называют *тестированием «белого ящика»* (*white box*) в противоположность классическому понятию «*черного ящика*» (*black box*).

При тестировании «черного ящика» программа рассматривается как объект, внутренняя структура которого неизвестна.

Тестировщик вводит данные и анализирует результат, но организация процесса обработки данных программой ему неизвестна. Подбирая тесты, специалист ищет интересные с его точки зрения входные данные и условия, которые могут привести к нестандартным результатам. Интересны для него, прежде всего те представители каждого класса входных данных, на которых с наибольшей вероятностью могут проявиться ошибки тестируемой программы.

При тестировании «стеклянного ящика» ситуация совершенно иная. Тестировщик (как правило, это программист) разрабатывает тесты, основываясь на знании исходного кода, к которому он имеет полный доступ. В результате он получает следующие преимущества:

направленность тестирования. Программист может тестировать программу по частям, разработать специальные тестовые подпрограммки, которые вызывают тестируемый модуль и передают ему интересующие программиста данные. Отдельный модуль легче протестировать именно как «стеклянный ящик»;

полный охват кода. Программист всегда может определить, какие именно фрагменты кода работают в каждом тесте. Он видит, какие еще ветви кода остались непротестированными и может подобрать условия, *в* которых они будут выполнены;

управление потоком. Программист всегда знает, какая функция должна выполняться в программе следующей и каким должно быть ее текущее состояние. Чтобы выяснить, работает ли программа так, как он думает, программист может включить в нее отладочные команды, отображающие информацию о ходе ее выполнения, или воспользоваться для этого специальным программным средством, называемым ***отладчиком.*** Отладчик может делать очень много полезных вещей: отслеживать и менять последовательность выполнения команд программы, показывать содержимое ее переменных и их адреса в памяти и др.;

отслеживание целостности данных. Программисту известно, какая часть программы должна изменять каждый элемент данных. Отслеживая состояние данных с помощью того же отладчика, он может выявить такие ошибки, как изменение дан-

ных не теми модулями, их неверная интерпретация или неудачная организация. Программист может и самостоятельно автоматизировать тестирование;

использование внутренних граничных точек. В исходном коде видны те граничные точки программы, которые скрыты от взгляда «извне». Например, для выполнения определенного действия может быть использовано несколько совершенно различных алгоритмов, и, не заглянув в код, невозможно определить, какой из них выбрал программист. Еще одним типичным примером может быть проблема переполнения буфера, используемого для временного хранения входных данных. Программист сразу может сказать, при каком количестве данных буфер переполнится, и ему не нужно при этом проводить тысячи тестов;

тестирование, определяемое выбранным алгоритмом. Для тестирования обработки данных, использующей очень сложные вычислительные алгоритмы, могут понадобиться специальные технологии. В качестве классических примеров можно привести преобразование матрицы и сортировку данных. Тестировщику нужно точно знать, какие алгоритмы используются, и в необходимых случаях использовать специальную литературу.

В данном случае тестирование «стеклянного ящика» рассматривается как часть процесса программирования. Программисты выполняют эту работу постоянно, они тестируют каждый модуль после его написания, а затем еще раз после интеграции его в систему. Этому их учат еще в учебных заведениях. В большинстве учебников по тестированию именно этому его виду отводится основная роль.

6.3. Регрессионное тестирование

Основной работой тестировщиков является регрессионное тестирование. У этого термина два значения, объединенных идеей повторного использования разработанных тестов. Предположим, что был проведен тест, в результате которого обнаружили ошибку, и программист ее исправил. После чего снова проводится тот же тест, чтобы убедиться, что ошибки больше нет. Это и есть регрессионное тестирование. Можно просмотреть несколько вариаций исходного теста, чтобы полностью проверить

исправленный фрагмент программы. В данном случае задача регрессионного тестирования состоит в том, чтобы убедиться, что выявленная ошибка полностью исправлена программистом и больше не проявляется.

В некоторых коллективах в набор регрессионных тестов включают каждую найденную ошибку, даже если она была исправлена ранее. Каждый раз, когда в программу вносится изменение, все тесты проводятся снова. Особенно важно провести такое обстоятельное тестирование, если программа изменяется спустя достаточно длительное время или изменения вносятся другим программистом. Исправления очень чувствительны к таким изменениям, поскольку в тексте программы, если только они не задокументированы самым тщательным образом, выглядят как непонятные или неудачные фрагменты.

Второй пример применения регрессионного тестирования. После выявления и исправления ошибки проводится стандартная серия тестов, но уже с другой целью — убедиться в том, что, исправляя одну часть программы, программист не испортил другую. В этом случае тестируется целостность всей программы, а не исправление одной ошибки.

Существует ***метод инкрементального тестирования*** основанный на регрессионных тестах с использованием разработанных заглушек и оболочек.

6.4. Тестирование «черного ящика»

6.4.1. Полный цикл тестирования разработанного программного продукта

Когда кодирование завершено, программа передается группе тестирования. Тестировщик ищет ошибки, составляет о них отчеты, затем получает новую версию программы и снова ищет ошибки. При этом находятся старые ошибки, не замеченные на первом этапе, и новые, появившиеся после доработки. В [2] подытожены данные об эффективности исправления найденных ошибок.

Если для исправления ошибки нужно изменить не более десяти операторов, вероятность того, что это будет сделано пра-

вильно с первого раза, составляет 50 %. Если для исправления ошибки нужно изменить около пятидесяти операторов, вероятность того, что это будет сделано правильно с первого раза, составляет 20 %.

Проблема состоит не только в том, что программист может не исправить ошибку до конца, а в большей степени в том, что исправления могут иметь *побочные эффекты.* Исправление одной ошибки может привести к появлению другой. Случается, что одна ошибка скрывает другую, которая проявляется только после ее устранения. К сожалению, программисты часто концентрируются только на поставленной перед ними проблеме и, решив ее, считают свою работу сделанной. Регрессионного тестирования, пусть даже самого поверхностного, они не выполняют.

Учитывая все сказанное, можно прийти к выводу, что тестировать одну и ту же программу приходится несколько раз. На ранних стадиях тестирования исправленные версии программы могут поступать каждые несколько часов или дней. Поэтому среди специалистов распространена практика не принимать новую версию, пока не будет самым тщательным образом протестирована предыдущая. Такое полное тестирование очередной версии с составлением итогового отчета обо всех известных проблемах и всех найденных в этой версии ошибках называют *полным циклом.*

Руководители проектов часто рассчитывают ограничиться двумя циклами тестирования: в первом выявить все ошибки, а во втором убедиться, что все они исправлены. Реально для тестирования может понадобиться порядка восьми циклов, а если на каждом из них тестировать программу менее тщательно — тогда от 20 до 30.

6.4.2. Стандартная процедура тестирования «черного ящика»

В этом разделе описывается стандартная последовательность событий, свойственная тестированию информационной системы как «черного ящика».

Планирование

Как и всякая работа, тестирование программного продукта начинается с планирования: определяется стратегия, разрабатываются серии тестов, распределяются между сотрудниками задания. Начинать планирование можно сразу после того, как команда проектировщиков выработает требования к программному продукту. Однако чаще подробное планирование начинается на первом цикле тестирования, когда к тестированию предоставляется готовый программный продукт.

Приемочное тестирование

При поступлении каждой новой версии информационной системы тестировщики, прежде всего, проверяют, достаточно ли она стабильна. Если система некорректно работает или даже «зависает» при проведении простейших тестов, следует прекратить тестирование и вернуть систему на доработку. Такое первое беглое тестирование называют *приемочным* или *квалификационным.*

Если приемочные тесты будут стандартизированы, их копии могут быть переданы программистам, чтобы те проводили их сами и не сдавали «сырые» программы на тестирование. Это позволит избежать возвратов тестировщиками неработающих программ, т. е. моментов, психологически неприятных для обеих сторон.

Приемочные тесты должны быть короткими. В них должны проверяться только основные функции и основные данные.

Во многих компаниях приемочные тесты частично автоматизированы с помощью специального программного обеспечения, выполняющего тестирование «черного ящика».

Проверка стабильности системы

Тестировщик должен оценить стабильность программы и количество циклов тестирования для составления календарного плана работ, оценить стоимость услуг по ее тестированию сторонним агентством или оценить качество программного продукта, который компания собирается купить и распространять.

В этом случае задача специалиста по тестированию состоит не в поиске ошибок, а в определении самых ненадежных частей программы. При этом начинать следует с изучения прилагаемого к программе руководства. В нем должны быть описаны все

функции программы и приведены простые и наглядные примеры. Кроме того, необходимо провести еще несколько тестов, на которых, по мнению тестировщика, программа должна сбоить. В конце такого оценочного тестирования должно сложиться определенное представление о работоспособности системы и трудоемкости ее тестирования: сколько в программе ошибок и насколько тяжело будет ее протестировать. К сожалению, на данный момент отсутствует механизм транслирования этого представление в человеко-часы. Тестировщику придется полагаться только на собственный опыт и профессиональный навык.

Обычно предварительная оценка стабильности программы занимает около недели. Если для тестирования всех ее функций этого времени недостаточно, необходимо проверить часть из них. При этом необходимо обязательно включить в отчет обзор каждого раздела руководства.

Таким образом, реально оценивая трудоемкость тестирования, можно прийти к выводу, что не стоит рассчитывать на то, что разобраться с системой удастся, например, за неделю, если только она не элементарна и если это не очередная версия продукта, который был уже несколько раз протестирован.

6.4.3. Тестирование производительности

Производительность является одной из важнейших характеристик современных информационных систем. Тестирование производительности, с одной стороны, позволяет найти узкие места и неоптимальные решения в системе и предложить методы их исправления. С другой стороны, тестирование производительности приложений на различных аппаратных комплексах может помочь заказчику подобрать необходимое оборудование, обеспечивающее требуемые показатели эффективности работы системы в целом.

При тестировании производительности системы применяются следующие тесты [19]:

собственно тестирование производительности — применение тестов, использующих постоянный объем нагрузки, на различные конфигурации системы и программного окружения, для определения приемлемости производительно-

сти тестируемой системы и ее настройки (или оптимизации). Измерению подлежит количество транзакций в минуту, число пользователей, размер используемой базы данных и т. п.;

сравнительное тестирование — применение тестов, использующих стандартные, рекомендованные нагрузки для измерения производительности системы и ее сравнения с рекомендованной производительностью системы (или измерениями);

нагрузочное тестирование — проверка и определение приемлемости действующих пределов системы при различных объемах нагрузки, при этом сама тестируемая система остается постоянной. Измеряются характеристики объема нагрузки и времени ответа;

анализ распределения и балансировки нагрузки. Если системы содержат распределенную архитектуру или балансировку нагрузок, проводятся специальные тесты, для проверки методов функций распределения и балансировки нагрузок;

тестирование конфликтов — проверка возможности системы корректно обрабатывать многочисленные запросы пользователей к одному ресурсу (записи данных, память, и т.п.);

тестирование объемов — проверка и анализ возможностей системы оперировать большими объемами данных. Проверяется как работа с большими объемами данных на входе\выходе системы, так и работа с внутренними данными системы;

стрессовое тестирование — проверка работы функций системы с некорректными параметрами. Стрессовое состояние системы может включать экстремальные нагрузки, дефицит памяти, отсутствие сервисов или аппаратного обеспечения, сжатые общие ресурсы;

тестирование на расширяемость — измерение и анализ скорости выполнения различных операций продукта на множестве конфигураций программного и аппаратного обеспечения и систем управления базами данных.

6.5. Завершающие этапы тестирования

Программный продукт должен пройти ряд завершающих тестов, после того как он готов и протестирован. Прежде всего, он должен быть еще раз сверен с наиболее подробными и близ-

кими к нему проектными документами. В частности, должно быть проведено **функциональное тестирование**, при котором продукт сверяется с внешней спецификацией.

Затем программа сверяется с опубликованной печатной документацией: сверка с пользовательскими требованиями — ***тестирование целостности***; сверка с системными требованиями — ***системное тестирование***. Эти две процедуры представляют собой так называемое **аттестационное тестирование.**

Бета-тестирование

Бета-тестирование представляет собой обратную связь с пользователями, которая осуществляется после подтверждения тестировщиком стабильности программы и наличия необходимой документации. На этом этапе с программой работают ее потенциальные пользователи. Они эксплуатируют программу и доводят до разработчика свои замечания. Поскольку бета-тестировщики знают, что в программе могут оставаться еще очень серьезные ошибки, они не работают с ней полный день — на 20 часов тестирования у них уходит около трех недель.

Однако в определенных ситуациях бета-тестировщики работают с продуктом гораздо более основательно. Такие ситуации возникают в следующих случаях:

- пользователю очень нужен разрабатываемый продукт, даже если он и не надежен, но других подобных продуктов на рынке нет;
- разработчик способен достойно «отблагодарить» тестировщика. Обычно в качестве платы за бета-тестирование пользователь получает или бесплатную копию продукта, или значительную скидку. Если продукт дорогой, этого достаточно. Но если речь идет о системе, предназначенной для доступа к важной информации, и в этой программе происходит сбой, потеря данных может обойтись пользователю гораздо дороже стоимости программного продукта;
- разработчик дает пользователю выгодную гарантию (например, дается обещание, что в случае сбоя программы сотрудником компании-разработчика будут бесплатно введены потерянные данные).

Тестирование целостности готового продукта

и тестирование распространяемых копий

С завершением тестирования последней бета-версии готового программного продукта возможные проблемы не заканчиваются. Разработчику необходимо убедиться, что система в целости и сохранности попадет к клиенту. Нередко на этом этапе случаются всевозможные неприятности: например, компании отсылают пользователям пустые или инфицированные диски.

Для тестирования распространяемых копий необходимо собирать все компоненты системы, которые будут отправлены пользователю, необходимо также проверить наличие документации. Только после этого продукт отправляется по назначению.

Например, комплект дисков проверить достаточно просто: достаточно выполнить их двоичное сравнение с последней версией продукта. Такое сравнение обойдется гораздо дешевле, чем отправка пользователям новой партии, если будет выявлено какое-либо несоответствие между задекларированной и отправленной версиями.

Настоятельно рекомендуется включить в эту процедуру тестирования также проверку на вирусы. Если программное обеспечение поставляется в сжатом виде, не следует останавливаться на тестировании сжатых файлов. Необходимо проверить компьютер на наличие вирусов какой-либо антивирусной программой: установив систему, запустить программу-антивирус и проверить распакованные файлы системы на наличие вирусов, после чего запустить систему и перезапустить компьютер и убедиться, что он не заражен.

Тестирование целостности программного продукта — работа более обстоятельная. Тестировщику желательно спрогнозировать все жалобы и критические замечания, которые могут поступить от пользователей в ближайшие несколько месяцев после внедрения системы. Этим может заниматься ведущий специалист, который не участвовал в данной разработке, или сотрудник независимого агентства. В его задачи не входит поиск ошибок, поскольку он предполагает, что и системное и функциональное тестирование проведены тщательным образом. Специалист внимательно сравнивает систему с пользовательской документацией и документами, в которых описаны требования к

системе. Кроме того, он может выполнить сравнение с конкурирующими продуктами.

В рамках тестирования целостности системы выполняется и анализ маркетинговых материалов. Возможности системы обязательно должны соответствовать рекламе. Потому и рекламная копия, и все печатные и электронные рекламные материалы перед публикацией должны подвергнуться проверке.

Тестирование целостности программного продукта лучше поручить конкретному человеку. Для однопользовательской системы средней сложности на это уйдет около двух недель.

Окончательная приемка и сертификация

Если ИС разрабатывается по контракту, то для ее приемки заказчик должен будет провести собственные тесты. Если система относится к классу простых систем, тестирование может быть неформальным. Однако для большинства проектов процедура приемочного тестирования заранее согласовывается и фиксируется в контрактных документах. Поэтому, прежде чем передать систему заказчику, необходимо убедиться, что она абсолютно безупречно проходит серию приемочных тестов. Обычно приемочное тестирование занимает не более одного дня и не является особенно тщательным.

Сертификация всегда выполняется сторонней фирмой — независимой или работающей на заказчика. Сертификационное тестирование может быть как коротким и поверхностным, так и более тщательным. По договору оно может выполняться вместо приемочного тестирования. При этом уровень сертификационного тестирования и стандарты, которым должны соответствовать система и процесс ее разработки, обязательно должны быть зафиксированы в договоре (контракте). Если сертификация проводится по желанию компании-разработчика, например из маркетинговых соображений, то разработчик сам определяет набор тестов.

Действительно эффектный тест должен быть достаточно простым и приводить к немедленному сбою. Как подбирать такие тесты, сказать сложнее. Здесь приходится основываться на своем собственном профессиональном опыте, знании слабых

мест программиста, операционной системы и результатах тестирования аналогичных продуктов.

6.6. Тестирование на этапе сопровождения

Значительная часть средств, которые организация-разработчик затрачивает на программный продукт, «уходит» уже после завершения его разработки — на этапе сопровождения программного продукта. На этом этапе также проводится тестирование программного продукта.

Адаптационное тестирование

Этот вид тестирования выполняется только на этапе сопровождения, когда система переносится с одной аппаратной или программной платформы на другую. Если система должна работать на нескольких типах компьютеров, необходимо проверить ее совместимость с каждым из них. Ниже представлено описание стратегии такого тестирования.

Проверка общего функционирования. При этом необходимо выполнить серию регрессионных тестов. Если какая-то из функций плохо совместима с новой платформой, то, скорее всего, произойдет сбой в ее работе и в процессе работы системы, так что тестировщику не стоит опасаться, что возникшие проблемы останутся незамеченными. Как правило, при переносе системы на другую платформу тесты на общее функционирование она проходит вполне успешно. Поэтому обычно такие тесты проводят в сжатые сроки.

Проверка клавиатуры. Если у компьютера специфическая клавиатура, в работе с ней могут быть небольшие отклонения от стандарта. Поэтому необходимо проверить нажатие каждой клавиши, и в различных комбинациях. Особое внимание здесь обращается на управляющие клавиши — <Shift>, <Alt> и т. п.

Проверка монитора. При проверке монитора проверяется, как система работает с новым терминалом, как отображается графика. Если программа работает в текстовом режиме, то все ли символы отображаются правильно, нет ли проблем с цветом, подчеркиванием или подсветкой.

Проверка номера версии и идентификация системы. Если номер версии системы изменился, необходимо убедиться, что

он нигде не остался старым. Если при запуске программа идентифицирует аппаратуру или операционную систему, необходимо убедиться, что она делает это правильно.

Проверка инсталляционного пакета. При установке программного продукта инсталляционной программе может потребоваться определение аппаратной и программной конфигурации системы. Необходимо убедиться, что она делает это правильно. При этом необходимо выполнить установку продукта в системах с различной конфигурацией, в сети и на отдельном компьютере, установить его поверх предыдущей версии.

Проверка совместимости. Предположим, что на исходном компьютере система была совместима с некой другой программой. Если эта программа также была перенесена на новый компьютер, необходимо проверить сохранилась ли совместимость.

Тестирование интерфейса. В различных графических средах (Windows, Mac, AmigaDOS, X-Win и т. п.) действуют различные соглашения о пользовательском интерфейсе. Перейдя в новую среду, система должна выглядеть в ней достаточно естественно.

Если программный продукт впервые адаптируется к новой платформе, тестирование может занять четверть того времени, которое было потрачено на разработку. Перенос на следующую платформу может пройти и быстрее, особенно если эти платформы достаточно совместимы.

Вернемся к вопросу о качестве. Если программный продукт создается по заказу конкретного клиента, тогда клиент может принимать в его проектировании самое непосредственное участие. Он предоставляет подробную спецификацию с описанием своих требований и собственного видения продукта, а разработчик соглашается все это реализовать. В таком случае качество будет означать точное соответствие спецификации клиента.

Нельзя забывать и о ***надежности системы.*** Надежность системы тем выше, чем реже в ней происходят сбои, особенно такие, которые влекут за собой потерю данных.

Хотя надежность программы исключительно важна, она не является единственным критерием ее качества. Если система не позволяет пользователю выполнить задекларированные функ-

ции, которые он считает важными, пользователь не будет ею доволен. А если пользователь недоволен работоспособностью системы, значит, качество программы нельзя назвать высоким.

Для тестировщика ***качество системы*** определяется:

– возможностями, благодаря которым система понравится пользователю;

– недостатками, которые вынуждают пользователя приобрести другую систему.

Главное, что тестировщик может сделать для улучшения качества системы, — это выявить ее недостатки, сбои в работе и явные ошибки. Ни надежность, ни функциональность программы не могут быть абсолютными, и ее качество в конечном счете означает разумный баланс между этими двумя характеристиками.

6.7. Организация и проведение испытаний на надежность

6.7.1. Цели и задачи проведения испытаний

При написании данного раздела использован материал, представленный в ГОСТ 16504—81, ГОСТ 19.004—80 и сведения, изложенные в [3, 20].

Кроме описанных выше методов тестирования АСОИУ проводятся **испытания системы**. Испытания в совокупности с тестированием являются важнейшим элементом управления качеством сложных АСОИУ, особенно если от их надежного функционирования зависят жизненно важные процессы (например, АСОИУ АЭС, банковские АСОИУ и др.). Испытание является завершающим этапом разработки.

В соответствии с ГОСТ 16504-81 под испытанием промышленной продукции понимают экспериментальное определение количественных и/или качественных характеристик объекта испытания в процессе его функционирования; а также при моделировании объекта и/или воздействии на него. Под испытанием программного обеспечения информационной системы следует понимать экспериментальное определение количественных и/или качественных характеристик свойств системы при ее функцио-

нировании в реальной среде и/или моделировании среды функционирования.

Целью испытания является экспериментальное определение фактических (достигнутых) характеристик свойств испытываемой АИС. Эти характеристики могут быть как количественными, так и качественными. Важно, чтобы на их основе можно было сделать вывод о пригодности данной системы к использованию по своему назначению. Если вывод отрицательный, то система возвращается на доработку.

Основными видами испытания АСОИУ являются предварительные, приемочные и эксплуатационные испытания, включающие опытную эксплуатацию. В зависимости от места проведения различают стендовые и полигонные испытания. Под испытательным стендом понимают совокупность технических устройств и математических моделей, обеспечивающих в автоматическом режиме имитацию среды функционирования; поступление входных данных; поступление искажающих воздействий; регистрацию информации о функционировании системы, а также управление процессом испытания и объектом испытания. Если в основу стендовых испытаний положен принцип моделирования, то соответствующие испытательные стенды называют моделирующими.

Испытательным полигоном называют место, предназначенное для испытаний в условиях, близких к условиям эксплуатации, и обеспеченное необходимыми средствами испытания. Полигонным испытаниям подвергают системы, работающие в реальном масштабе времени. В полигонных условиях обычно сочетают натурные испытания с использованием реальных объектов автоматизируемых систем и моделирование некоторых объектов и процессов их функционирования.

По степени зависимости испытателей от разработчиков различают зависимые и независимые испытания. При зависимых испытаниях основные операции с испытываемыми АСОИУ (подготовка к работе, подготовка и ввод исходных данных, регистрация и анализ результатов) выполняют тестировщики организации исполнителя. Оценку результатов испытания производит комиссия при активном участии разработчиков. Незави-

симые испытания проводят специальные подразделения, не несущие ответственности за разработку программ и непосредственно не подчиняющиеся руководителям разработки.

6.7.2. Технологическая схема испытания

Для повышения эффективности испытания, его ускорения и удешевления необходимо разработать научно обоснованные методы, средства и методики, позволяющие преодолеть недостатки подхода к испытанию. Эта цель может быть достигнута лишь путем разработки технологической схемы испытаний, которая предусматривает:

- знание назначения испытываемого ПС, условий его функционирования и требований к нему со стороны пользователей;
- автоматизацию всех наиболее трудоемких процессов и прежде всего моделирование среды функционирования, включая искажающие воздействия;
- ясное представление цели и последовательности испытания;
- целенаправленность и неизбыточность испытания, исключающие или минимизирующие повторение однородных процедур при одних и тех же условиях функционирования испытываемой АСОИУ;
- систематический контроль за ходом, регулярное ведение протокола и журнала испытания;
- четкое, последовательное определение и исполнение плана испытания;
- четкое сопоставление имеющихся ресурсов с предполагаемым объемом испытания;
- возможность обеспечения полноты и достоверности результатов испытания на всех этапах, а также их объективной количественной оценки.

Любому виду испытаний должна предшествовать тщательная подготовка. В подготовку испытаний АСОИУ входят следующие мероприятия:

- составление и согласование плана-графика проведения испытания;

- разработка, комплектование, испытание и паспортизация программно-технических средств, используемых при испытаниях;
- анализ пригодности испытательных средств, используемых во время предварительных испытаний, для проведения приемочных испытаний;
- анализ пригодности накопленных данных о качестве системы для использования при окончательном определении значений показателей качества испытываемой системы;
- проверка и согласование с представителем Заказчика документации на систему, предъявляемой при испытаниях;
- разработка, согласование и утверждение программ и методик испытаний;
- аттестация специалистов на допуск к проведению испытаний;
- приемка испытываемого опытного образца системы на носителе данных и документации;
- проведение мероприятий, направленных на обеспечение достоверности испытаний.

Особо следует подчеркнуть необходимость заблаговременной разработки и испытания всех программно-технических средств, которые будут использоваться при проведении испытаний. При этом следует иметь в виду, что уровень точности и надежности измерительной аппаратуры, используемой при испытаниях любого объекта, должен быть значительно выше соответствующих показателей испытываемого объекта. Поэтому реальные характеристики программно-технических испытательных средств необходимо установить заранее, а их приемлемость согласовывать между разработчиками, испытателями и заказчиками системы.

Выделяется пять этапов испытаний:

1) обследование проектируемой АСОИУ, анализ проектной документации;

2) определение наиболее важных подсистем и функций проектируемой системы, подлежащих испытанию;

3) анализ показателей качества системы и методов определения их значений. Разработка программ и методик испытания;

4) разработка (освоение) испытательных программно-технических средств, библиотек тестов и баз данных в необходимых случаях;

5) непосредственное проведение испытаний, анализ результатов, принятие решения.

В зависимости от специфики, условий применения, требований к качеству испытываемых систем испытания могут проводиться либо путем тестирования, либо путем статистического моделирования среды функционирования, либо на основе натурных и смешанных экспериментов. Часто полезно использование всех этих методов. Значения некоторых показателей качества можно получить экспертным путем.

6.7.3. Планирование и оценка завершенности испытаний

План проведения испытаний должен быть ориентирован на обеспечение всесторонней проверки систем и максимальной (заданной) достоверности полученных результатов при использовании ограниченных ресурсов, выделенных на испытаниях. Принципиально возможны следующие подходы к решению этой задачи:

1) анализ специалистами всего диапазона входных данных. На основе анализа заранее готовят такое множество комбинаций данных (тестовых наборов данных), которое охватывает наиболее характерные подмножества входных данных. Здесь возможно использование методов, используемых при тестировании «черного ящика»;

2) анализ специалистами множества ситуаций, которые могут возникнуть при функционировании АСОИУ. Выбирают наиболее характерные ситуации. Каждую из них выражают через тестовый набор входных данных. Далее сущность испытания и анализа результатов сводится к первому подходу;

3) испытание системы в реальной среде функционирования;

4) испытание системы в статистически моделируемой среде функционирования, адекватной реальной среде.

При этом следует заметить, что ни один из этих подходов не является универсальным. Каждый из них имеет свои преимущества и недостатки, которые в разной степени проявляются

в зависимости от специфики испытываемой системы. Наиболее достоверные результаты получаются при испытаниях в реальной среде функционирования. Но такие испытания не всегда удается осуществить. Поэтому на практике используют комбинации всех видов

Анализ показывает, что абсолютная проверка систем ни при одном из рассмотренных подходов не осуществима. Поэтому при планировании испытаний необходимо предварительно анализировать структуры испытываемых программ и входных данных.

В общем случае при планировании и организации испытаний следует искать компромиссное решение, учитывающее два противоречивых требования: обеспечение максимальной достоверности обобщенной оценки качества системы и выполнение испытания в ограниченное время с использованием ограниченных ресурсов.

Следует выделить три стадии испытания: подготовительную; непосредственные испытания; заключительную (подготовка отчетных материалов).

Между выделенными стадиями испытания системы имеются прямые и обратные связи, аналогичные связям между этапами жизненного цикла АСОИУ. Это означает, что при выполнении работ заключительной стадии может быть выявлена необходимость возвращения к стадии непосредственных испытаний (или даже к подготовительной стадии) для уточнения отдельных характеристик.

Испытания считаются завершенными, и система считается прошедшей испытания, если в ходе последнего цикла испытаний специалистами не было выявлено ни одной ошибки.

6.7.4. Автоматизация проведения испытаний и процесса тестирования

Автоматизация процесса проведения испытаний и тестирования увеличивает количество тестов, которые могут быть выполнены за ограниченный промежуток времени. Это может быть достигнуто с учетом следующих требований:

1) системы автоматической аттестации, которые генерируют данные для тестов, передают или преобразуют данные и результаты тестов и оценивают результат, должны вести полный протокол своих действий;

2) соответствующие системы автоматизированного тестирования должны использоваться для тестирования и/или симуляции поведения исполняемых кодов целевой системы;

3) системы автоматизированного тестирования должны использоваться с целью обеспечения гарантии и подтверждения корректности загрузки выполняемый кода системы.

В ряде случаев может быть использован следующий перечень вспомогательных программ:

- генераторы тестов, программ анализа тестового покрытия и тестовых драйверов;
- программы диагностики реального времени с контролем аварийных записей программы, и свойствами отслеживания;
- отладчики с возможностями отладки на уровне исходного кода;
- наборы автоматических тестов для облегчения регрессивного тестирования.

При автоматизированном тестировании могут быть использованы специальные аналитические системы, способные проверять спецификации, проекты и коды. К таким системам относятся:

1) ***программа контроля синтаксиса***, предоставляющая информацию о структуре программы, использовании данных, зависимости выходных переменных от входных и результатах контроля технологического процесса, с помощью реализации следующих функций:

- выявление дефектов структуры, таких как множественные одновременные запуски, множественные выходы, недоступный код, избыточный код и не использование результатов выполнения каких-либо функций;
- определение иерархии модулей/процедур;
- выявление противоречий стандартов и правил программирования, включая проверки неправильных скобок в циклах;

• идентификация данных, которые читаются до их записи, записывающихся до чтения, записанных дважды без промежуточного чтения;

• проверка соответствия потока информации спецификациям;

• помощь в проектировании плана динамических тестов;

• управление данными теста и возможными сгенерированными данными теста.

2) ***программа проверки семантики,*** описывающая математические взаимосвязи между выходными и входными переменными для каждого семантически правдоподобного пути с помощью ветвления свободных сегментов программы. Это позволяет контролировать действия программы в различных состояниях и выявлять ошибки, такие как наличие выходных значений, несоответствующих входным данным, ошибочная реакция на не ожидавшиеся входные значения, несоответствие функций и операторов и т. д.;

3) ***генератор формальной проверки.*** Формальная проверка проекта требует использования интерактивных программ, выполняющих необходимые символьные манипуляции под руководством тестировщика для снятия проверки обязательств. Программы проверки доказательств должны основываться на формализованно доказанной теории. Данные программы необходимо проверить на соответствие этой доказанной теории;

4) ***программа анализа соответствия***, показывающую корректность реализации в программе технических требований. Такие программы в доказательствах используют предварительные условия и условия, накладываемые после реализации обратных постоянных циклов. Эти системы периодически подтверждают реализацию каждого условия в коде.

6.8. Анализ и интерпретация результатов тестирования

В процессе проведения тестирования, а также по завершении каждого из этапов проводится анализ и интерпретация результатов тестирования. По итогам проделанной работы тестировщиком предоставляется достаточно подробная и наглядная документация, где кроме описания результатов тестирования

приводится анализ дефектов и ошибок, описываются возможные причины их появления и методы их исправления или устранения.

Кроме того, по результатам тестирования производительности приводятся рекомендации по возможным методам ее повышения, а также рекомендации по аппаратно-техническим средствам, использование которых позволит получить систему с необходимым уровнем эффективности (производительности). Содержание отчета о ходе тестирования приводится в п. 5.4.2. настоящего пособия.

6.9. Программные ошибки

Определение ошибок

Приступая к тестированию системы, специалист должен иметь представление о том, какого типа ошибки он может обнаружить. В [2] даются следующие определения ошибок:

1) ***программная ошибка*** **—** ситуация, когда программа не выполняет действий, которых пользователь от вполне обоснованно ожидает;

2) ***программная ошибка*** **—** это исключительно субъективная характеристика, показывающая, насколько программа не справляется со своей задачей.

Не существует ни абсолютного определения ошибок, ни точного критерия наличия их в программе. Заметим, что второе определение не касается таких явных ошибок, как ошибки вычислений по точно известным формулам. Бывает довольно сложно убедить программиста, что недостаток пользовательского интерфейса является ошибкой или что эта ошибка очень серьезна или даже в том, что тестировщик вообще имеет право заниматься такими вопросами. Но пользователи обращают внимание на подобные ошибки не меньше, чем на очевидные сбои в работе системы. В этом разделе описываются категории программных ошибок, охватывающих большинство возможных ошибок в программном обеспечении.

Ошибки обработчика ошибок

Процедуры обработки ошибок — это очень важная часть программного обеспечения системы. Однако и в них достаточно часто встречаются ошибки. Кроме того, правильно определив

ошибку, система не всегда выдает о ней достаточно информативное сообщение. Таким образом, при написании программного кода системы, необходимо с особой тщательностью подходить к созданию обработчика ошибок.

Ошибки, связанные с обработкой граничных условий

Простейшими граничными условиями являются числовые, однако существуют и другие граничные ситуации. Любой аспект работы системы, к которому применимы понятия больше или меньше, раньше или позже, первый или последний, короче или длиннее, обязательно должен быть проверен на границах диапазона. Внутри диапазонов программа обычно работает достаточно корректно, однако на их границах могут быть замечены отклонения либо ошибки.

Ошибки вычислений

Программирование даже простых арифметических операций чревато ошибками. Одной из распространенных среди математических ошибок являются ошибки округления. После нескольких промежуточных вычислений может оказаться, что 2 + 2 = –1, даже если на промежуточных этапах не было логических ошибок. К этой категории относятся и ошибки, вызванные неправильным выбором алгоритма. Сюда можно отнести неправильные формулы, формулы, неприменимые к обрабатываемым данным, неверные способы разбиения сложных выражений на более простые элементы. В случае алгоритмической ошибки код в точности выполняет то, что имел в виду программист, — он правильно закодировал неверную идею.

Ошибки начального и последующего состояния

В ходе работы системы может возникнуть ситуация, когда при выполнении какой-либо функции системы сбой происходит только однажды — при первом обращении к этой функции. На экране может появиться искаженное изображение или странная информация. Возможно, что в этом случае неверно выполнятся расчеты, запустятся бесконечные циклы или операционная система выдаст сообщение о нехватке памяти. Причиной такого по-ведения программы может быть отсутствие файла с инициализационной информацией. После первого же запуска программа создаст такой файл, и дальше все будет в порядке. Полу-

чается, что такую ошибку невозможно повторить (точнее, ее повтор может быть только в случае установки новой копии программы). Однако данная ошибка может испортить впечатление пользователя о всей системе в целом.

Иногда, программируя процесс, связанный с последовательными преобразованиями информации, разработчики забывают о том, что у пользователя может возникнуть необходимость в возврате исходных данных и их замене. При этом необходимо учесть следующие аспекты:

- корректность поведения системы в данной ситуации;
- возможность внесения в систему необходимых изменений и вероятность сохранности выполненной пользователем работы;
- корректность представленной на экране информации при возвращении к исходному состоянию программы: свои данные или стандартные значения, которыми программа инициализирует переменные при запуске.

Ошибки управления потоком

Если по логике программы вслед за первым действием должно быть выполнено второе, а она выполняет третье, значит, в управлении потоком допущена ошибка. Такие ошибки трудно пропустить, поскольку в большинстве случаев в работе системы произойдет так называемый мягкий сбой.

Ошибки передачи или интерпретации данных

Один модуль системы может передавать данные другому модулю или другой программе. Некоторые данные могут передаваться между модулями множество раз, и на каком-то этапе эти данные могут быть разрушены или неверно интерпретированы при дальнейшей работе системы. При этом следует учесть, что изменения, внесенные одной из частей программы, могут потеряться или достичь не всех частей системы, где должна быть проведена их дальнейшая обработка. Выявлена такая ошибка может только при тщательной проработке программы тестирования взаимодействия программных модулей.

Ошибка приоритетов («ситуация гонок»)

Классическая ситуация гонок описывается так. Предположим, в системе ожидаются два события *A* и *B*. Первым может произойти любое из них. Однако, если первым произойдет со-

бытие *A*, выполнение программы продолжится, а если первым наступит событие *B*, то произойдет сбой. Программист полагал, что первым всегда должно быть событие *A*, и не ожидал, что событие *B* может произойти раньше. Тестировать ситуации гонок довольно сложно. Наиболее типичны они для систем, где параллельно выполняются взаимодействующие процессы и потоки, а также для многопользовательских систем реального времени. Ошибки в таких системах трудно воспроизвести, и на их выявление обычно требуется очень много времени.

Ошибки, возникающие вследствие перегрузки

Система может не справляться с повышенными нагрузками (например, не выдерживать интенсивной и длительной эксплуатации или не справляться со слишком большими объемами данных). Кроме того, сбои в системе могут происходить из-за недостаточного объема памяти как на рабочей станции, так и на сервере данных, или из-за отсутствия других необходимых ресурсов. У каждой системы существуют свои пределы, при этом возникает вопрос о соответствии реальных возможностей и требований программы к ресурсам ее спецификации и поведении программы при перегрузках.

Ошибки, связанные с работой аппаратного обеспечения

Система может посылать устройствам неверные данные, игнорировать их сообщения об ошибках, пытаться использовать устройства, которые заняты или вообще отсутствуют. Даже если нужное устройство просто сломано, программа должна понять это и сообщить пользователю, а не «сбоить» при попытке к нему обратиться.

В случае, если в состав АСОИУ входит некоторое аппаратное обеспечение, необходима тщательная проверка надежности его функционирования в различных эксплуатационных условиях.

Ошибки версий системы

В ходе эксплуатации системы могут возникнуть ситуации, когда старые ошибки вдруг всплывают снова из-за того, что программа скомпонована с устаревшей версией одной из подпрограмм. Поэтому версии всех составляющих проекта обязательно должны централизованно контролироваться. Кроме того, следует убедиться, что правильны все появляющиеся на экране

сообщения об авторских правах, названии и номере версии программного продукта. Обязательной проверке подлежат все копии программного продукта. Обычно контроль версий программы и ее исходного кода осуществляется группой контроля качества (Quality Assurance).

Ошибки документации

Сама по себе документация не является программным обеспечением системы, однако, это часть программного продукта. Если документация написана неквалифицированно, не соответствует текущей версии системы, не полностью охватывает все составляющие системы, то пользователь может прийти к выводу, что и сама система не намного лучше. Для создания качественной документации к ее разработке необходимо привлекать высококвалифицированных технических писателей.

Ошибки тестирования

Специалист по тестированию, как и программист, может допускать ошибки в ходе проведения тестов. Обнаружение ошибок, допущенных тестировщиками, периодически происходит, и их количество зависит от квалификации тестировщика и от качества и полноты программы тестирования. При этом следует учесть, что иногда ошибки тестировщика отражают проблемы пользовательского интерфейса: если система заставляет пользователя делать ошибки, значит, систему нужно подвергнуть повторному тестированию и провести эргономическую экспертизу. Большинство же ошибок тестирования вызваны неверными тестовыми данными.

Ошибки пользовательского интерфейса

С программой может быть трудно (или даже невозможно) работать по множеству причин. Такие ошибки можно объединить под названием «ошибки пользовательского интерфейса». Ниже представлены несколько разновидностей таких ошибок.

Ошибки функциональности

Функциональные недостатки имеют место, если информационная система не делает того, чего ожидает от нее пользователь, выполняет одну из своих функций плохо или не полностью. Хотя функции системы достаточно подробно описываются в ее спецификации, окончательное представление о том, что

программа должна делать, существует только в представлении ее пользователей. Функциональные недостатки обнаруживаются абсолютно у всех программ, поскольку ожидания пользователей — вещь субъективная: у разных пользователей представления о функциях одной и той же системы различны.

Однако во многих случаях функциональный недостаток вполне очевиден. Если предусмотренную программой задачу трудно выполнить, если она решается плохо или при определенных обстоятельствах вообще не может быть решена – необходимо пересмотреть функциональную составляющую системы. И если ожидания пользователей вполне разумны и достаточно обоснованны, эту проблему можно назвать ошибкой.

Взаимодействие программы с пользователем

В процессе взаимодействия программы с пользователем важно учесть следующие аспекты:

- сложность ориентирования пользователя в программе и способ представления справочной информации;
- наличие в системе экранных инструкций и подсказок, степень их информативности;
- наличие в системе интерактивных справок и возможность для пользователя в случае затруднений найти в них реальную помощь;
- корректность выдаваемых пользователю сообщений системы об ошибках и способе их исправления;
- наличие в системе элементов, которые могут раздражать пользователя;

Организация программы

Здесь важно определить, насколько легко пользователю «потеряться» в программе. Нет ли в ней непонятных команд или команд, которые достаточно легко спутать между собой? Какие ошибки чаще всего делает пользователь, на что он тратит больше всего времени и почему?

Пропущенные команды

Следует определить, каких операций или функций система не выполняет, не заставляет ли программа выполнять некоторые действия странным, неестественным или крайне неэффективным способом. При этом следует понять, нельзя ли привести ее

в соответствие с привычным стилем работы пользователя. В ряде случаев можно добавить возможность настройки функций.

Производительность

В интерактивном программном обеспечении очень важна скорость. Недопустимо, если у пользователя создается впечатление, что система работает медленно, если он *чувствует* задержки в ее реакции на выполнение команд (особенно если конкурирующие системы в схожих ситуациях могут работать значительно быстрее).

Выходные данные

Большинство программ, так или иначе, формируют выходные данные: отображают информацию на экране, печатают ее или сохраняют в файлах. Следует определить, получает ли пользователь то, что хочет увидеть. Правильно ли формируются отчеты, наглядны ли диаграммы и достаточно ли отчетливо они выглядят на бумаге? Сохраняются ли данные в формате, доступном и для других аналогичных программ? Обладает ли программа достаточной гибкостью, чтобы можно было подстраивать ее под нужды конкретного пользователя?

Перечисленные выше ошибки могут иметь место в любой автоматизированной информационной системе обработки информации и управления, и в задачи всех участников разработки входит минимизация их появления в ходе эксплуатации информационной системы конечным пользователем.

Контрольные вопросы

1. Опишите процессы тестирования на всех этапах создания информационных систем.
2. Поясните методику тестирование «стеклянного ящика».
3. Поясните методику тестирование «черного ящика».
4. Перечислите и кратко охарактеризуйте ошибки, возникающие в процессе функционирования системы.

Литература

1. Сборник временных норм на работы по ведению Государственного мониторинга геологической среды, информационной деятельности, цифровому картографированию [Электронный ресурс]: публикация сайта / Томск: ТомскГеомониторинг, 2001. — Режим доступа к сайту: http://www.tgm.ru

2. Канер Сэм. Тестирование программного обеспечения: пер. с англ. / Сэм Канер, Джек Фолк, Енг Кек Нгуен. — К.: Изд-во «ДиаСофт», 2000. — 544 с.

3. Липаев В. В. Надежность программных средств. Серия «Информатизация России на пороге XXI века». — М.: СИНТЕГ, 1998. — 232 с.

4. Гнеденко Б. В. Математические методы в теории надежности / Б.В. Гнеденко, Ю.К. Беляев, А.Д. Соловьев. — М.: Наука. Физматгиз , 1965, 524 с.

5. Саймон А. Р. Стратегические технологии баз данных: менеджмент на 2000 год / А.Р. Саймон. — М.: Финансы и статистика, 1999. — 479 с.

6. Хубаев Г. Н. Сравнение сложных программных систем по критерию функциональной полноты / Г.Н. Хубаев // Программные продукты и системы. — 1998. — № 2. — С. 6–9.

7. Павлов А. Н. Обзор систем документооборота [Электронный ресурс]: публикация сайта / М.: Электронные офисные системы, 2000. — Режим доступа к сайту: http://www.documenta.ru/uslugi/model.html. — Заглавие с экрана

8. Электронные офисные системы [Электронный ресурс]: публикация сайта / М.: Электронные офисные системы, 2005. — Режим доступа к сайту: http://www.eos.ru/razl_delo.html. — Заглавие с экрана

9. Электронная канцелярия. Золушка-Win. Служебная корреспонденция НТЦ Институт развития Москвы: Руководство пользователя. — М., 2001. — 60 с.

10. Электронная канцелярия, описание применения [Электронный ресурс]: публикация сайта / М.: Группа компаний «Системы и проекты», 2000. — Режим доступа к сайту: http://www.mdi.ru/dis/new/disclass.html. — Заглавие с экрана

11. Примеры использования систем автоматизированного делопроизводства и электронного документооборота [Электронный ресурс]: публикация сайта / М.: Microsoft, 2000. — Режим доступа к сайту: http://www.microsoft.com/rus/government/docflow. — Заглавие с экрана

12. Дружинин Г. В. Надежность автоматизированных систем / Г.В. Дружинин. — М.: Энергия, 1977. — 170 с.

13. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем / А.М. Вендров. — М.: Финансы и статистика, 1998. —

14. Клюев М. Н., О дизайне/Эргономика... [Электронный ресурс]: публикация сайта / М.: 2000. — Режим доступа к сайту: http://www.rosdesign.com/index.htm

15. Дизайн пользовательского интерфейса [Электронный ресурс]: публикация сайта / М.: — Isys Information Architects Inc, 2000. — Режим доступа к сайту: http://www.akzhan.midi.ru

16. Рекомендации по пользовательскому интерфейсу» [Электронный ресурс]: публикация сайта / М.: Microsoft, 2000. — Режим доступа к сайту: http://www.Microsoft.com/rus. — Заглавие с экрана

17. Жоголев Е. А. Лекции по технологии программирования [Электронный ресурс]: публикация сайта / М.: МГУ, 2002. — Режим доступа к сайту: http://sp.cmc.msu.ru/info/3/techprog.htm

18. Дружинин Г. В. Стандартизация в области качества служебной информации / Г.В. Дружинин, И.В. Сергеева // Стандарты и качество. — 2003. — № 10. — С. 27–29.

19. Тестирование и исследования[Электронный ресурс]: публикация сайта / М.: Amphora Group, 2004. — Режим доступа к сайту: http://www.aqt.ru/services.htm

20. Майерс, Гленфорд Дж. Надежность программного обеспечения: Пер. с англ./ Г. Майерс; Пер. Ю. Ю. Галимов, Ред. В. Ш. Кауфман. — М.: Мир, 1980. — 360 с.

Приложение

ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

Дата введения с 01.01.1990г.

Настоящий стандарт распространяется на автоматизированные системы (АС) для автоматизации различных видов деятельности (управление, проектирование, исследование и т. п.), включая их сочетания, и устанавливает состав, содержание, правила оформления документа «Техническое задание на создание (развитие или модернизацию) системы» (далее - ТЗ на АС).

Рекомендуемый порядок разработки, согласования и утверждения ТЗ на АС приведен в приложении 1.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. ТЗ на АС является основным документом, определяющим требования и порядок создания (развития или модернизации - далее создания) автоматизированной системы, в соответствии с которым проводится разработка АС и ее приемка при вводе в действие.

1.2. ТЗ на АС разрабатывают на систему в целом, предназначенную для работы самостоятельно или в составе другой системы.

Дополнительно могут быть разработаны ТЗ на части АС:

- на подсистемы АС, комплексы задач АС и т. п. в соответствии с требованиями настоящего стандарта;
- на комплектующие средства технического обеспечения и программно-технические комплексы в соответствии со стандартами ЕСКД и СРПП;
- на программные средства в соответствии со стандартами ЕСПД;
- на информационные изделия в соответствии с ГОСТ 19.201 и НТД, действующей в ведомстве заказчика АС.

Примечание. В ТЗ на АСУ для группы взаимосвязанных объектов следует включать только общие для группы объектов требования.

Специфические требования отдельного объекта управления следует отражать в ТЗ на АСУ этого объекта.

1.3. Требования к АС в объеме, установленном настоящим стандартом, могут быть включены в задание на проектирование вновь создаваемого объекта автоматизации. В этом случае ТЗ на АС не разрабатывают.

1.4. Включаемые в ТЗ на АС требования должны соответствовать современному уровню развития науки и техники и не уступать аналогичным требованиям, предъявляемым к лучшим современным отечественным и зарубежным аналогам. Задаваемые в ТЗ на АС требования не должны ограничивать разработчика системы в поиске и реализации наиболее эффективных технических, технико-экономических и других решений.

1.5. ТЗ на АС разрабатывают на основании исходных данных, в том числе содержащихся в итоговой документации стадии «Исследование и обоснование создания АС», установленной ГОСТ 24.601.

1.6. В ТЗ на АС включают только те требования, которые дополняют требования к системам данного вида (АСУ, САПР, АСНИ и т. д.), содержащиеся в действующих НТД, и определяются спецификой конкретного объекта, для которого создается система.

1.7. Изменения к ТЗ на АС оформляют дополнением или подписанным заказчиком и разработчиком протоколом. Дополнение или указанный протокол являются неотъемлемой частью ТЗ на АС. На титульном листе ТЗ на АС должна быть запись «Действует с ... ».

2. СОСТАВ И СОДЕРЖАНИЕ

2.1. ТЗ на АС содержит следующие разделы, которые могут быть разделены на подразделы:

- общие сведения;
- назначение и цели создания (развития) системы;
- характеристика объектов автоматизации;
- требования к системе;
- состав и содержание работ по созданию системы;

- порядок контроля и приемки системы;
- требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
- требования к документированию;
- источники разработки.

В ТЗ на АС могут включаться приложения.

2.2. В зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ.

В ТЗ на части системы не включают разделы, дублирующие содержание разделов ТЗ на АС в целом.

2.3. В разделе «Общие сведения» указывают:

- полное наименование системы и ее условное обозначение;
- шифр темы или шифр (номер) договора;
- наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты;
- перечень документов, на основании которых создается система, кем и когда утверждены эти документы;
- плановые сроки начала и окончания работы по созданию системы;
- сведения об источниках и порядке финансирования работ;
- порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы.

2.4. Раздел «Назначение и цели создания (развития) системы» состоит из подразделов:

- назначение системы;
- цели создания системы.

2.4.1. В подразделе «Назначение системы» указывают вид автоматизируемой деятельности (управление, проектирование и т. п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать.

Для АСУ дополнительно указывают перечень автоматизируемых органов (пунктов) управления и управляемых объектов.

2.4.2. В подразделе «Цели создания системы» приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания АС, и указывают критерии оценки достижения целей создания системы.

2.5. В разделе «Характеристики объекта автоматизации» приводят:

- краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию;
- сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

Примечание: Для САПР в разделе дополнительно приводят основные параметры и характеристики объектов проектирования.

2.6. Раздел «Требования к системе» состоит из следующих подразделов:

- требования к системе в целом;
- требования к функциям (задачам), выполняемым системой;
- требования к видам обеспечения.

Состав требований к системе, включаемых в данный раздел ТЗ на АС, устанавливают в зависимости от вида, назначения, специфических особенностей и условий функционирования конкретной системы. В каждом подразделе приводят ссылки на действующие НТД, определяющие требования к системам соответствующего вида.

2.6.1. В подразделе «Требования к системе в целом» указывают:

- требования к структуре и функционированию системы;
- требования к численности и квалификации персонала системы и режиму его работы;
- показатели назначения;
- требования к надежности;
- требования безопасности;
- требования к эргономике и технической эстетике;
- требования к транспортабельности для подвижных АС;
- требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
- требования к защите информации от несанкционированного доступа;
- требования по сохранности информации при авариях;
- требования к защите от влияния внешних воздействий;
- требования к патентной чистоте;
- требования по стандартизации и унификации;
- дополнительные требования.

2.6.1.1. В требованиях к структуре и функционированию системы приводят:

- 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 создаваемой системы со смежными системами, требования к ее совместимости, в том числе указания о способах обмена информацией (автоматически, пересылкой документов, по телефону и т. п.);
- требования к режимам функционирования системы;
- требования по диагностированию системы;
- перспективы развития, модернизации системы.

2.6.1.2. В требованиях к численности и квалификации персонала на АС приводят:

- требования к численности персонала (пользователей) АС;

- требования к квалификации персонала, порядку его подготовки и контроля знаний и навыков;
- требуемый режим работы персонала АС.

2.6.1.3. В требованиях к показателям назначения АС приводят значения параметров, характеризующие степень соответствия системы ее назначению.

Для АСУ указывают:

- степень приспособляемости системы к изменению процессов и методов управления, к отклонениям параметров объекта управления;
- допустимые пределы модернизации и развития системы;
- вероятностно-временные характеристики, при которых сохраняется целевое назначение системы.

2.6.1.4. В требования к надежности включают:

- состав и количественные значения показателей надежности для системы в целом или ее подсистем;
- перечень аварийных ситуаций, по которым должны быть регламентированы требования к надежности, и значения соответствующих показателей;
- требования к надежности технических средств и программного обеспечения;
- требования к методам оценки и контроля показателей надежности на разных стадиях создания системы в соответствии с действующими нормативно-техническими документами.

2.6.1.5. В требования по безопасности включают требования по обеспечению безопасности при монтаже, наладке, эксплуатации, обслуживании и ремонте технических средств системы (защита от воздействий электрического тока, электромагнитных полей, акустических шумов и т. п.), по допустимым уровням освещенности, вибрационных и шумовых нагрузок.

2.6.1.6. В требования по эргономике и технической эстетике включают показатели АС, задающие необходимое качество взаимодействия человека с машиной и комфортность условий работы персонала.

2.6.1.7. Для подвижных АС в требования к транспортабельности включают конструктивные требования, обеспечивающие транспортабельность технических средств системы, а также требования к транспортным средствам.

2.6.1.8. В требования к эксплуатации, техническому обслуживанию, ремонту и хранению включают:

- условия и регламент (режим) эксплуатации, которые должны обеспечивать использование технических средств (ТС) системы с заданными техническими показателями, в том числе виды и периодичность обслуживания ТС системы или допустимость работы без обслуживания;
- предварительные требования к допустимым площадям для размещения персонала и ТС системы, к параметрам сетей энергоснабжения и т. п.;
- требования по количеству, квалификации обслуживающего персонала и режимам его работы;
- требования к составу, размещению и условиям хранения комплекта запасных изделий и приборов;
- требования к регламенту обслуживания.

2.6.9. В требования к защите информации от несанкционированного доступа включают требования, установленные в НТД, действующей в отрасли (ведомстве) заказчика.

2.6.1.10. В требованиях по сохранности информации приводят перечень событий: аварий, отказов технических средств (в том числе - потеря питания) и т. п., при которых должна быть обеспечена сохранность информации в системе.

2.6.1.11. В требованиях к средствам защиты от внешних воздействий приводят:

- требования к радиоэлектронной защите средств АС;

- требования по стойкости, устойчивости и прочности к внешним воздействиям (среде применения).

2.6.1.12. В требованиях по патентной чистоте указывают перечень стран, в отношении которых должна быть обеспечена патентная чистота системы и ее частей.

2.6.1.13. В требования к стандартизации и унификации включают: показатели, устанавливающие требуемую степень использования стандартных, унифицированных методов реализации функций (задач) системы, поставляемых программных средств, типовых математических методов и моделей, типовых проектных решений, унифицированных форм управленческих документов, установленных ГОСТ 6.10.1, общесоюзных классификаторов технико-экономической информации и классификаторов других категорий в соответствии с областью их применения, требования к использованию типовых автоматизированных рабочих мест, компонентов и комплексов.

2.6.1.14. В дополнительные требования включают:

- требования к оснащению системы устройствами для обучения персонала (тренажерами, другими устройствами аналогичного назначения) и документацией на них;
- требования к сервисной аппаратуре, стендам для проверки элементов системы;
- требования к системе, связанные с особыми условиями эксплуатации;
- специальные требования по усмотрению разработчика или заказчика системы.

2.6.2. В подразделе «Требование к функциям (задачам)», выполняемым системой, приводят:

 по каждой подсистеме перечень функций, задач или их комплексов (в том числе обеспечивающих взаимодействие частей системы), подлежащих автоматизации;

при создании системы в две или более очереди - перечень функциональных подсистем, отдельных функций или задач, вводимых в действие в 1-й и последующих очередях;

 временной регламент реализации каждой функции, задачи (или комплекса задач);
 требования к качеству реализации каждой функции (задачи или комплекса задач), к форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов;
 перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

2.6.3. В подразделе «Требования к видам обеспечения» в зависимости от вида системы приводят требования к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другие видам обеспечения системы.

2.6.3.1. Для математического обеспечения системы приводят требования к составу, области применения (ограничения) и способам, использования в системе математических методов и моделей, типовых алгоритмов и алгоритмов, подлежащих разработке.

2.6.3.2. Для информационного обеспечения системы приводят требования:

- к составу, структуре и способам организации данных в системе;
- к информационному обмену между компонентами системы;
- к информационной совместимости со смежными системами;
- по использованию общесоюзных и зарегистрированных республиканских, отраслевых классификаторов, унифицированных документов и классификаторов, действующих на данном предприятии;
- по применению систем управления базами данных;
- к структуре процесса сбора, обработки, передачи данных в системе и представлению данных;

- к защите данных от разрушений при авариях и сбоях в электропитании системы;
- к контролю, хранению, обновлению и восстановлению данных;
- к процедуре придания юридической силы документам, продуцируемым техническими средствами АС (в соответствии с ГОСТ 6.10.4).

2.6.3.3. Для лингвистического обеспечения системы приводят требования к применению в системе языков программирования высокого уровня, языков взаимодействия пользователей и технических средств системы, а также требования к кодированию и декодированию данных, к языкам ввода-вывода данных, языкам манипулирования данными, средствам описания предметной области (объекта автоматизации), к способам организации диалога.

2.6.3.4. Для программного обеспечения системы приводят перечень покупных программных средств, а также требования:

- к независимости программных средств от используемых СВТ и операционной среды;
- к качеству программных средств, а также к способам его обеспечения и контроля;
- по необходимости согласования вновь разрабатываемых программных средств с фондом алгоритмов и программ.

2.6.3.5. Для технического обеспечения системы приводят требования:

- к видам технических средств, в том числе к видам комплексов технических средств, программно-технических комплексов и других комплектующих изделий, допустимых к использованию в системе;
- к функциональным, конструктивным и эксплуатационным характеристикам средств технического обеспечения системы.

2.6.3.6. В требованиях к метрологическому обеспечению приводят:

- предварительный перечень измерительных каналов;

- требования к точности измерений параметров и (или) к метрологическим характеристикам измерительных каналов;
- требования к метрологической совместимости технических средств системы;
- перечень управляющих и вычислительных каналов системы, для которых необходимо оценивать точностные характеристики;
- требования к метрологическому обеспечению технических и программных средств, входящих в состав измерительных каналов системы, средств, встроенного контроля, метрологической пригодности измерительных каналов и средств измерений, используемых при наладке и испытаниях системы;
- вид метрологической аттестации (государственная или ведомственная) с указанием порядка ее выполнения и организаций, проводящих аттестацию.

2.6.3.7. Для организационного обеспечения приводят требования:

 к структуре и функциям подразделений, участвующих в функционировании системы или обеспечивающих эксплуатацию;
 к организации функционирования системы и порядку взаимодействия персонала АС и персонала объекта автоматизации;
 к защите от ошибочных действий персонала системы.

2.6.3.8. Для методического обеспечения САПР приводят требования к составу нормативно-технической документации системы (перечень применяемых при ее функционировании стандартов, нормативов, методик и т. п.).

2.7. Раздел «Состав и содержание работ по созданию (развитию) системы» должен содержать перечень стадий и этапов работ по созданию системы в соответствии с ГОСТ 24.601, сроки их выполнения, перечень организаций - исполнителей работ, ссылки на документы, подтверждающие согласие этих организаций на участие в создании системы, или запись, определяющую ответственного (заказчик или разработчик) за проведение этих работ.

В данном разделе также приводят:

- перечень документов, по ГОСТ 34.201, предъявляемых по окончании соответствующих стадий и этапов работ;
- вид и порядок проведения экспертизы технической документации (стадия, этап, объем проверяемой документации, организация-эксперт);
- программу работ, направленных на обеспечение требуемого уровня надежности разрабатываемой системы (при необходимости);
- перечень работ по метрологическому обеспечению на всех стадиях создания системы с указанием их сроков выполнения и организаций-исполнителей (при необходимости).

2.8. В разделе «Порядок контроля и приемки системы» указывают:

- виды, состав, объем и методы испытаний системы и ее составных частей (виды испытаний в соответствии с действующими нормами, распространяющимися на разрабатываемую систему);
- общие требования к приемке работ по стадиям (перечень участвующих предприятий и организаций, место и сроки проведения), порядок согласования и утверждения приемочной документации;
- статус приемочной комиссии (государственная, межведомственная, ведомственная).

2.9. В разделе «Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие» необходимо привести перечень основных мероприятий и их исполнителей, которые следует выполнить при подготовке объекта автоматизации к вводу АС в действие.

В перечень основных мероприятий включают:

- приведение поступающей в систему информации (в соответствии с требованиями к информационному и лингвистическому обеспечению) к виду, пригодному для обработки с помощью ЭВМ;
- изменения, которые необходимо осуществить в объекте автоматизации;

- создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ;
- создание необходимых для функционирования системы подразделений и служб;
- сроки и порядок комплектования штатов и обучения персонала.

Например, для АСУ приводят:

- изменения применяемых методов управления;
- создание условий для работы компонентов АСУ, при которых гарантируется соответствие системы требованиям, содержащимся в ТЗ.

2.10. В разделе «Требования к документированию» приводят:

- согласованный разработчиком и Заказчиком системы перечень подлежащих разработке комплектов и видов документов, соответствующих требованиям ГОСТ 34.201 и НТД отрасли заказчика;
  перечень документов, выпускаемых на машинных носителях; требования к микрофильмированию документации;
- требования по документированию комплектующих элементов межотраслевого применения в соответствии с требованиями ЕСКД и ЕСПД;
- при отсутствии государственных стандартов, определяющих требования к документированию элементов системы, дополнительно включают требования к составу и содержанию таких документов.

2.11. В разделе «Источники разработки» должны быть перечислены документы и информационные материалы (технико-экономическое обоснование, отчеты о законченных научно-исследовательских работах, информационные материалы на отечественные, зарубежные системы-аналоги и др.), на основании которых разрабатывалось ТЗ и которые должны быть использованы при создании системы.

2.12. В состав ТЗ на АС при наличии утвержденных методик включают приложения, содержащие:

- расчет ожидаемой эффективности системы;
- оценку научно-технического уровня системы.

Приложения включают в состав ТЗ на АС по согласованию между разработчиком и заказчиком системы.

3. ПРАВИЛА ОФОРМЛЕНИЯ

3.1. Разделы и подразделы ТЗ на АС должны быть размещены в порядке, установленном в разд. 2 настоящего стандарта.

3.2. ТЗ на АС оформляют в соответствии с требованиями ГОСТ 2.105 на листах формата А4 по ГОСТ 2.301 без рамки, основной надписи и дополнительных граф к ней.

Номера листов (страниц) проставляют, начиная с первого листа, следующего за титульным листом, в верхней части листа (над текстом, посередине) после обозначения кода ТЗ на АС.

3.3. Значения показателей, норм и требований указывают, как правило, с предельными отклонениями или максимальным и минимальным значениями. Если эти показатели, нормы, требования однозначно регламентированы НТД, в ТЗ на АС следует приводить ссылку на эти документы или их разделы, а также дополнительные требования, учитывающие особенности создаваемой системы. Если конкретные значения показателей, норм и требований не могут быть установлены в процессе разработки ТЗ на АС, в нем следует сделать запись о порядке установления и согласования этих показателей, норм и требований:

«Окончательное требование (значение) уточняется в процессе ...
и согласовывается протоколом с ... на стадии ...».

При этом в текст ТЗ на АС изменений не вносят.

3.4. На титульном листе помещают подписи заказчика, разработчика и согласующих организаций, которые скрепляют гербовой печатью. При

необходимости титульный лист оформляют на нескольких страницах. Подписи разработчиков ТЗ на АС и должностных лиц, участвующих в согласовании и рассмотрении проекта ТЗ на АС, помещают на последнем листе.

Форма титульного листа ТЗ на АС приведена в приложении 2. Форма последнего листа ТЗ на АС приведена в приложении 3.

3.5. При необходимости на титульном листе ТЗ на АС допускается помещать установленные в отрасли коды, например: гриф секретности, код работы, регистрационный номер ТЗ и др.

3.6. Титульный лист дополнения к ТЗ на АС оформляют аналогично титульному листу технического задания. Вместо наименования «Техническое задание» пишут «Дополнение № ... к ТЗ на АС ... ».

3.7. На последующих листах дополнения к ТЗ на АС помещают основание для изменения, содержание изменения и ссылки на документы, в соответствии с которыми вносятся эти изменения.

3.8. При изложении текста дополнения к ТЗ следует указывать номера соответствующих пунктов, подпунктов, таблиц основного ТЗ на АС и т. п. и применять слова: «заменить», «дополнить», «исключить», «изложить в новой редакции».

ФОРМА ТИТУЛЬНОГО ЛИСТА ТЗ НА АС

_____

.

наименование организации - разработчика ТЗ на АС

УТВЕРЖДАЮ
Руководитель (должность, наименование предприятия - заказчика АС)
Личная подпись Расшифровка подписи
Печать
Дата

УТВЕРЖДАЮ
Руководитель (должность, наименование предприятия - разработчик" АС)
Личная подпись Расшифровка подписи
Печать
Дата

_____
наименование вида АС

_____
наименование объекта автоматизации

_____
сокращенное наименование АС

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
На ____ листах

Действует с

СОГЛАСОВАНО

Руководитель (должность, наименование согласующей организации)
Личная подпись Расшифровка подписи
Печать
Дата

ФОРМА ПОСЛЕДНЕГО ЛИСТА ТЗ НА АС

(код ТЗ)

СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия имя, отчество	Подпись	Дата

СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия имя, отчество	Подпись	Дата