Институт системного программирования
Российской академии наук

В.В. Липаев

ТЕСТИРОВАНИЕ КОМПОНЕНТОВ И КОМПЛЕКСОВ ПРОГРАММ

УЧЕБНИК

СИНТЕГ®
Москва - 2010

УДК 004.41(075.8)
ББК 32.973.26-018я73
Л61

Липаев В.В.
Тестирование компонентов и комплексов программ. Учебник.
– М.: СИНТЕГ, 2010. – 400 с.
ISBN 978-5-89638-115-0

Учебник состоит из двух частей.

В первой части (7 лекций) рассматриваются системные основы разработки требований к сложным комплексам программ, эталоны при их проектировании и производстве, декомпозиция функций и архитектуры комплексов программ для формирования требований к компонентам и модулям. Изложены требования к характеристикам качества, к тестам и допустимым рискам комплексов программ (КП).

Во второй части (7 лекций) представлены методы тестирования потоков управления и потоков данных программных модулей. Рассмотрено планирование тестирования модулей и компонентов для КП, нисходящая – восходящая сборка и тестирование программных компонентов, подготовка и применение графиков разработки и исполнения тестов для компонентов и комплексов программ. Изложены организация и процессы испытаний, Программа и методики тестирования компонентов и сложных комплексов программ.

Учебник ориентирован на специалистов, студентов и аспирантов для обучения тестированию модулей и программных компонентов, а также крупных программных комплексов высокого качества

Рецензенты:
А.К. Петренко – д.ф.-м.н, зав.отделом технологии программирования
ИСП РАН
Б.М. Позднеев – д.т.н., профессор, проректор по информатизации
МГТУ «СТАНКИН»

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В стране быстро увеличивается общий объем производства программных продуктов и сложность отдельных проектов. Это является одной из причин, возникновения проблем при их разработке, как и то, что многие предприятия, занимающиеся их производством, не уделяют должного внимания автоматизации и обеспечению всего жизненного цикла крупных программных комплексов, а также эффективному применению современных методов управления такими проектами и международным стандартам. Ориентация на программные продукты, поступающие на рынок с Запада, не может решить многие специфические проблемы конкретного создания и применения программных продуктов в стране. Для многих заказчиков и пользователей систем возникла острая необходимость поиска и выбора квалифицированных и надежных отечественных специалистов-подрядчиков, способных создавать сложные программные средства и базы данных высокого качества в разумные сроки, с учетом ограничений на затраты труда и другие ресурсы. Обострилась проблема собственных, отечественных разработок оригинальных крупных программных продуктов для ряда отраслей государственного управления, народного хозяйства и оборонной техники. Эти проблемы ***негативно отражаются на экономике, качестве и конкурентоспособности отечественных информационных систем***.

Крупные комплексы программ являются важнейшими, ***интеллектуальными компонентами систем***, реализующими обычно их основные, функциональные свойства. Методология управления и развития программных комплексов зависит от многих факторов, от квалификации специалистов, технических, организационных и договорных требований и сложности проектов. Множество текущих состояний и модификаций компонентов в жизненном цикле крупных программных комплексов, менеджерам необходимо упорядочивать, контролировать их развитие и реализацию участниками проекта. Организованное, контролируемое и методичное отслеживание динамики изменений программ и данных, их разработка при строгом учете и контроле каждого изменения, является ***базовой проблемой эффективного, поступательного развития*** каждой крупной информаци-

онной системы. Накопление в мире знаний, опыта разработки и применения большого количества различных компонентов и сложных комплексов программ способствовало систематизации и обобщению методов и технологий их разработки, сокращению дефектов и неопределенностей в характеристиках и качестве поставляемых и применяемых программных продуктов. В результате сформировалась ***современная методология и инженерная дисциплина*** обеспечения процессов жизненного цикла сложных программных продуктов – ***программная инженерия*** для различных областей применения.

Основная проблема управления созданием крупных программных комплексов – сводить воедино усилия многих исполнителей – специалистов разной квалификации, чтобы они выступали при производстве компонентов и всего комплекса как ***«команда»***, а не как разрозненная группа независимых, функциональных специалистов. В результате должны обеспечиваться ***концептуальная целостность систем*** и высокое качество решения главных задач и функций ***при сбалансированном использовании ресурсов***.

Программные продукты все чаще встраиваются в различные сложные системы реального времени. Работа над такими проектами требует от руководителей и программных инженеров, широкого взгляда и освоения ***общих методов проектирования и применения систем*** определенного назначения и сферы использования. Проблему рационального сочетания целей, стратегии действий, конкретных процедур и доступных ресурсов, необходимо решать для достижения ***основной цели получения программного комплекса с заданными функциональными характеристиками и качеством***. Базой эффективного управления производством сложного программного комплекса является ***план***, в котором задачи исполнителей частных работ и компонентов должны быть согласованы между собой по результатам и срокам их достижения, а также с выделяемыми для них ресурсами. Планирование проектирования и производства комплекса программ должно обеспечивать компромисс между требующимися функциями и характеристиками создаваемой системы и ограниченными ресурсами, необходимыми на ее разработку и применение.

Возросла роль ***регламентирования производства и интеграции*** модулей и компонентов, применения соответствующих методов и инструментария программной инженерии. Для обеспечения высокого качества программных продуктов особенно важно упорядочен-

ное, планируемое тестирование их модулей, компонентов и комплексов программ в целом. Руководителям тестирования необходимо участвовать в разработке требований для всей системы, а также осваивать прикладную область применения создаваемого комплекса программ до начала тестирования и обдумывания функций компонентов, их характеристик и тестов, требованиям которых должен отвечать программный продукт. Однако вследствие принципиальной новизны и сложности этих задач, они трудно воспринимаются традиционными программистами и тестировщиками модулей и небольших компонентов и множеством преподавателей отечественной высшей школы. Коренные отличия между методами и инструментарием индивидуального программирования и тестирования небольших программ и технологией планомерной, инженерной разработки и тестирования ***модулей, компонентов и сложных программных комплексов*** приводят к тому, что последние медленно осваиваются и входят в практику работы над крупными проектами больших коллективов специалистов.

Тестирование программ – основной метод, способный обеспечивать и удостоверять ***необходимое качество*** компонентов и комплексов программ, выявлять и устранять в них дефекты и ошибки при проектировании и производстве. Технологически тестирование ***имеет две цели:*** первая состоит в поиске, обнаружении и устранении дефектов и ошибок в программах и в применении покрытия тестами требований к программному комплексу; вторая определяется возможностью статистической оценки того, что при тестировании не проявится дефект или ошибка компонента или комплекса программ. Обе интерпретации этих целей одинаково важны для тестирования и достижения высокого качества программ. Тестирование для поиска, идентификации и устранения дефектов подразумевает успешность процедуры тестирования, если ***дефект найден***. Это отличается от подхода в тестировании, когда тесты исполняются для демонстрации того, что компонент или программный комплекс полностью удовлетворяет предъявляемым требованиями и, соответственно, тест считается успешным, если ***не найдено дефектов***. Тестирование программ может использоваться для демонстрации наличия дефектов, но никогда не гарантирует их отсутствие. Основная причина этого в том, что ***полное, всеобъемлющее тестирование недостижимо*** для реальных сложных программных комплексов. Когда оцениваются и применяются методы тестирования, надо четко определять, что подразумева-

ется под их эффективностью, желательно, в количественных величинах. Только обладая такого рода данными можно говорить о корректности сравнения и оценки эффективности разных методов тестирования.

Основная ***цель тестирования комплекса программ и его функциональных компонентов*** состоит в том, чтобы обнаруживать, регистрировать и устранять дефекты и ошибки, которые внесены во время последовательной разработки и реализации требований к его функциям и характеристикам. Тестированию программных модулей и компонентов посвящена обширная литература, однако в ней мало внимания уделяется тестированию сложных комплексов программ на соответствие исходным требованиям к их функциям. Учебник ориентирован на освоение специалистами методов ***последовательного тестирования модулей, компонентов и программных комплексов***, которые совместно должны подготавливать и обеспечивать высокое качество сложных программных продуктов. Общая ***структура учебника*** построена по традиционной **V** – схеме, в которой левая часть буквы отражает процессы разработки и конкретизации требований к комплексу, компонентам и модулям программ сверху вниз, а правая часть представляет процессы последовательного тестирования снизу вверх, начиная от модулей и до полного комплекса программ. Соответственно, весь цикл лекций представлен в виде двух частей:

- ***проектирование и разработка требований*** к функциям и характеристикам программных комплексов, компонентов и модулей – часть 1;
- ***тестирование и обеспечение корректного функционирования*** модулей, компонентов и комплексов программ – часть 2.

Реализация частных требований при разработке модулей и компонентов комплексов программ, а также их тестирование считаются исходными для сборки программных комплексов, в соответствии с исходными требованиями к их функциям и характеристикам. Для идентификации, локализации и устранения основной массы дефектов и ошибок, необходимо применять последовательное тестирование и корректировку модулей, компонентов и комплексов программ с использованием соответствующих методов и средств. Организованное, контролируемое и методичное отслеживание изменений компонентов в комплексах программ, их разработка при строгом учете и контроле каждого изменения, является ***базовой задачей*** эффективного, посту-

пательного развития сложных программных продуктов высокого качества. Исходными для этого развития всегда являются: цели, функции, характеристики и допустимые риски реализации требований, которые в цикле лекций иерархически представляются для ***объектов трех уровней: комплексов программ, компонентов и модулей***. Внутренняя структура процессов тестирования для этих объектов в основном подобны и включают: формирование требований к функциям и характеристикам; тестирование на соответствие требованиям к объекту; обнаружение и идентификацию дефектов и ошибок; корректировку и устранение дефектов; испытания и удостоверении корректности объектов, и документирование результатов тестирования. Они имеют ***следующие особенности***:

- ***комплексы программ*** зависят от сложности функциональных задач системы, могут содержать сотни и более компонентов и входить, как составные части, в более сложные комплексы, что повышает трудности формализации требований и определяет необходимость ***наиболее высокой квалификации*** уникальных руководителей, аналитиков и специалистов для их тестирования;
- ***компоненты*** могут состоять из десятков и сотен модулей, для разработки и тестирования которых необходимы высококвалифицированные специалисты аналитики, программисты, тестировщики и интеграторы, чья квалификация и деятельность трудно формализуются, сильно зависят от взаимодействия ***множества разных специалистов в «командах»*** для создания программных компонентов;
- ***модули*** наиболее многочисленные элементы комплексов программ, которые разрабатываются ***наименее квалифицированными специалистами*** тестировщиками и программистами, чьи результаты в совокупности характеризуются наибольшим количеством относительно простых дефектов и ошибок, вследствие чего формализации их тестирования в лекциях уделено значительное внимание.

Требования к квалификации специалистов для создания этих объектов высокого качества значительно различаются. Практически во всех лекциях приходится касаться особенностей необходимой их квалификации для решения соответствующих функциональных задач. Основное внимание сосредоточено на методах и формализации полноты и качества процессов тестирования для всех трех типов объектов. В лекциях представленные объекты не всегда четко разделены и формализованы, некоторые процессы, особенно связанные с тестированием модулей и компонентов, рассматриваются совместно. Они

использованы в учебнике как ***базовые элементы*** при анализе методов тестирования в жизненном цикле комплексов программ. Это определило следующие ***основные допущения и особенности*** изложения ***тестирования программных комплексов на соответствие эталонам и требованиям***:

- ***объектами тестирования*** являются сложные комплексы программ (сотни тысяч строк) для систем обработки информации и управления в реальном времени, разрабатываемые большими коллективами – «командами» специалистов, создающими ***программные компоненты и модули высокого качества***;
- в результате тестирования программные комплексы должны соответствовать ***целостному комплексу требований*** (***первому эталону***) – к функциям, характеристикам, архитектуре и качеству, утвержденному заказчиком и согласованному с разработчиками;
- наборы ***применяемых совокупностей тестов*** рассматриваются как ***второй эталон*** функций и характеристик модулей, компонентов и программных комплексов, которые должны адекватно отражать и покрывать весь набор требований к конкретному программному эталону соответствующего уровня;
- в качестве ***третьего эталона*** для ряда комплексов программ используется ***эксплуатационная документация***, которая должна обеспечивать эффективное применение программного продукта пользователями в соответствии с исходными требованиями к функциям, характеристикам и допустимым рискам;
- разработка требований и тестирование модулей, компонентов и комплексов программ осуществляется методами программной инженерии в условиях ***регулярного планирования и управления*** обнаружением и устранением дефектов и ошибок (отладкой), а также при ***ограниченных ресурсах***, выделяемых заказчиком на весь их жизненный цикл;
- для тестирования модулей, компонентов и комплексов программ применяются сценарии и процедуры ***детерминированных*** и некоторых типов ***динамических тестов***, на основе аттестованных моделей внешней среды и/или систем реального времени;
- процессы и результаты формирования требований и реализации тестирования модулей, компонентов и комплексов программ должны быть снабжены глубоким, детальным ***документированием,***

архивированием и ***конфигурационным управлением*** исходными, промежуточными и отчетными документами.

Требования к программному продукту должны быть зафиксированы в ***соглашении*** (договоре и техническом задании) между заказчиком и выполняющими проект руководителями и специалистами, отражающем потребности заказчика и пользователей в таком виде, чтобы разработчики могли построить удовлетворяющий их комплекс программ, его компоненты и модули. Требования должны являться, ***конкретным исходным эталоном*** при разработке комплекса программ, чтобы можно было при тестировании и заключительных испытаниях установить, когда они удовлетворены полностью или в какой степени. Эти данные должны устанавливать состав, содержание и значения результатов исполнения программ, которые следует получать системе или пользователям при определенных условиях и исходных данных.

Тесты и спецификации требований к ним являются вторыми эталонами представления требований к модулям, компонентам и комплексу программ. При разработке необходимо иметь пригодные для применения тестирования исходные ***эталоны – требования и/или сценарии использования функций*** системы. Их необходимо анализировать и определять в терминах адекватных требований к функциям и характеристикам программного комплекса, а также ***к тестам***, устанавливающим их содержание и корректность. Такая вторая форма описаний содержания программного комплекса, компонентов и модулей должна формироваться и использоваться для сквозной верификации спецификаций требований к тестам сверху вниз, а также подвергаться верификации на корректность соответствия исходным требованиям к комплексу, компонентам и модулям программ разного уровня.

Особенности ***деятельности специалистов*** при разработке требований к функциям, процедурам и характеристикам исполнения программ, а также описания и реализация ***программистами*** требований к модулям, компонентам и комплексам программ, существенно отличаются от представлений и методов описания тех же функций программ ***тестировщиками*** – создателями сценариев и процедур тестирования результатов их исполнения. Они акцентируют свою деятельность на конкретных требованиях к процедурам проверки функционирования, возможных результатах и взаимодействии компонентов программных комплексов. Это позволяет выявлять дефек-

ты, и повышать качество путем ***сопоставления двух методов*** и результатов описания одних и тех же функций и характеристик определенных программ. При этом существенно, что мала вероятность одинаковых ошибок в требованиях к сценариям и результатам исполнения тестов и в требованиях к функциям и характеристикам исполнения модулей, компонентов и комплексов программ. Кроме того, параллельная и независимая разработка спецификаций требований к функциям программ и спецификаций требований к адекватным тестам позволяет распараллеливать по разным специалистам производственные работы, что ведет к сокращению сроков создания модулей, компонентов и комплексов программ необходимого качества.

По существу, требования к комплексу (модулю или компоненту) программ и тесты для проверки их адекватности и полноты ***отражают один и тот же объект***, но в разной форме. Поэтому сложность представительного описания тестов соизмерима со сложностью описания полной совокупности требований к функциям, характеристикам и соответствующего текста программ. Вследствие этого трудоемкость и другие ***ресурсы*** для разработки адекватных тестов ***должны соответствовать*** ресурсам и трудоемкости при создании и реализации программистами корректных требований, определяющих полностью тестируемый объект. Таким образом, программной (процедурной) форме представления текстов программ должно полностью соответствовать его ***равноправное содержание*** в форме сценариев и значений тестов для проверки их взаимного соответствия. При этом ***дефекты и ошибки возможны в обеих формах*** описания комплекса программ (не только в текстах программного комплекса, но и в содержании тестов), определение их места и устранение является основной задачей тестирования. Однако на практике обычно это не учитывается, и выделяемые на тестирование ресурсы бывают значительно меньше и недостаточными для полноценного тестирования требований к модулям, компонентам и сложным комплексам программ, что определяет их не полное соответствие требованиям и недостаточное качество.

Основная цель создания программного комплекса состоит в обеспечении его применения пользователями или в системе, в соответствии с назначением и ***эксплуатационной документацией*** или документами, необходимыми для корректного функционирования системы. Эта документация должна быть адекватна требованиям к

функциям и характеристикам программного продукта, однако она разрабатывается обычно ***независимыми специалистами*** на основе исходных требований и их представлений о способах применения продукта. В результате документация может не полностью соответствовать исходным требованиям заказчика и функциям, которые проверены при тестировании, однако эксплуатационные характеристики, доступные для пользователей должны рассматриваться как конечная цель всего проекта. С этой позиции эксплуатационные документы являются еще одним, ***третьим эталоном для тестирования***, которому должен соответствовать программный комплекс. В процессе такого тестирования могут выявляться дефекты, ошибки и противоречия: в требованиях заказчика, в программном комплексе и компонентах, в результатах их тестирования, а также в эксплуатационных процедурах и документах. Эти недостатки должны устраняться в процессах системного анализа для достижения взаимной ***адекватности всех трех представлений*** необходимых функций и характеристик программного продукта.

Понятие ***«тестирование»*** далее используется в широком смысле этого слова, включая (не явно) устранение обнаруженных дефектов и ошибок. При этом внимание акцентируется на методах и процессах выявления дефектов и ошибок при тестировании. Особые специалисты и работы требуются для ***«отладки»*** комплексов программ, корректировки, устранения ошибок и обеспечения их соответствия исходным требованиям к заданным функциям и качеству. Для этого необходима локализация ошибок, их диагностика, разработка корректировок программ, контроль выполненных изменений и ***регрессионное тестирование*** для регистрации результатов устранения обнаруженных ошибок. Процессы корректировки являются особенно сложными и требуют участия в них соответствующих программистов модулей, иногда не только тех, где обнаружены ошибки, а также возможно руководителей разработки компонентов и может быть всего комплекса программ. Тем самым процессы ***выходят из сферы тестирования*** в область разработки текстов программ, для чего необходимы специалисты, участвующие в создании всего комплекса программ. Эти процессы определяются функциональными и специфическими особенностями всего производства комплексов программ и в учебнике не рассматриваются. После того, как проведены корректировки программ, приходится возвращаться к деятельности тестировщиков для ***регрессионного их тестирования*** и удостоверения, что

действительно устранены обнаруженные ошибки и при этом не проявились новые.

Современные программные продукты, поставляемые заказчикам или на рынок, должны иметь конкретные, гарантированные и документированные цели, функции и характеристики. В то же время в жизненном цикле сложных комплексов программ невозможно в начале проекта предусмотреть все требования к функциям, характеристикам и качеству абсолютно точно, без дефектов и ошибок. Кроме того, программные комплексы обычно имеют длительный жизненный цикл с рядом версий, в течение которого они совершенствуются, расширяются, и повышается их качество. Поэтому процессы формирования изменений и реализации требований должны организовываться на ***основе регулируемых итераций версий***, в которых периоды поставляемых стабильных, испытанных версий применения программных продуктов чередуются с интервалами пошагового расширения функций, совершенствования качества и оперативного адаптирования к характеристикам внешней среды у конкретных пользователей или в системах. Для этого в сложных проектах целесообразно применять международные стандарты, регламентирующие жизненный цикл комплексов программ (см. Приложение 1), и ***жесткую дисциплину*** координируемой деятельности коллектива специалистов для производства модулей, компонентов и комплексов программ, и документируемого ***конфигурационного управления версиями*** требований и тестов для обеспечения качества реализации изменяющихся требований.

Для решения важнейших проблем развития и применения современных систем требуется ***подготовка и воспитание квалифицированных специалистов в области индустрии производства сложных программных продуктов высокого качества***. Необходимо их обучение методам и современной программистской культуре промышленного создания высококачественных продуктов. Они должны уметь формализовать требования и достигать при тестировании необходимые значения качества функционирования и применения сложных комплексов программ, с учетом ограниченных ресурсов, которые доступны для получения и совершенствования этого качества. В жизненном цикле сложных программных средств для обеспечения их высокого качества целесообразно ***выделять специалистов***, ответственных за соблюдение регламентированной технологии проектиро-

вания и производства, за измерение и контроль качества комплексов программ в целом и их компонентов. Необходимо обучать этих специалистов анализу и оцениванию конкретных факторов, влияющих на качество функционирования программных продуктов со стороны реально существующих и потенциально возможных дефектов и ошибок в программах.

Учебник ориентирован на специалистов, студентов и аспирантов, имеющих знания и опыт программирования, тестирования модулей и программных компонентов, пригодных для использования в сложных комплексах программ. Он может служить методической базой для учебного курса, способствующего подготовке специалистов для ***разработки и тестирования сложных программных комплексов*** с учетом рекомендаций международных стандартов. Учебник полезно использовать при создании инструкций и практических руководств на предприятиях, реализующих сложные программные продукты, требующие применения методов программной инженерии. Основные рисунки учебника подготовлены в форме таблиц, отражающих структуру и содержание лекций, которые на практике при их чтении целесообразно использовать путем предварительного ***формирования слайдов***, например, с помощью системы «Презентация – Microsoft Power Point».

Часть 1

РАЗРАБОТКА ТРЕБОВАНИЙ К КОМПЛЕКСАМ ПРОГРАММ И КОМПОНЕНТАМ

Лекция 1.1

ОРГАНИЗАЦИЯ ТЕСТИРОВАНИЯ КОМПОНЕНТОВ И КОМПЛЕКСОВ ПРОГРАММ

Уровни организации тестирования комплексов программ (TMM)

Эффективность процессов тестирования в значительной степени зависят от уровня организации, применяемых методов и технологических средств разработки и тестирования модулей, компонентов и комплексов программ – рис. 1.1. Их можно отражать ***моделью зрелости тестирования (TMM)*** (Test Maturity Model), разработанной Институтом технологии программного обеспечения США. Она содержит набор уровней зрелости организации, через которые проходит предприятие с целью усовершенствования процессов тестирования. Это способствует повышению профессионализма при тестировании программных комплексов по аналогии с моделью технологической зрелости (**CMMI**) для жизненного цикла сложных комплексов программ, которая разработана Институтом программной инженерии США (SEI). Предприятия, заинтересованные в оценке и улучшении своих возможностей по тестированию, включаются в общее усовершенствование процесса разработки программ. Наличие однозначно соответствующих уровней в обеих моделях зрелости логически упростит параллельное усовершенствование процессов. Тестирование – это часть всего процесса разработки программных комплексов, следовательно, рост его зрелости нуждается в поддержке областей ключевых процессов, связанных с ростом эффективности основных процессов производства. Любая организация, желающая усовершенствовать свой процесс тестирования с помощью внедрения TMM,

должна, прежде всего, заняться усовершенствованием процессов жизненного цикла программ в целом, определив основные направления CMMI.

Организация тестирования компонентов и комплексов программ должны включать:

- определение уровня организации тестирования комплексов программ (TMM);
- модель организации процессов тестирования модулей, компонентов и комплексов программ;
- организацию и руководителей коллектива специалистов для тестирования программных комплексов;
- установление источников дефектов и ошибок в компонентах и комплексах программ:
  - понятие дефекта или ошибки в программе;
  - статистика и характеристики ошибок и дефектов в компонентах и комплексах программ;
  - устранение первичных ошибок, на основе их вторичных проявлений;
- типы системных дефектов и ошибок в сложных комплексах программ:
  - организационные дефекты требований и эталонов к программному комплексу;
  - системные ошибки и недостатки определения требований к программному комплексу;
  - ошибки определения характеристик системы и внешней среды;
  - дефекты и ошибки программных комплексов, которые проявляются как риски;
  - ошибки комплексов программ по сложности обнаружения и масштабу корректировок;
  - сложность проявления, обнаружения и устранения ошибок;
  - ошибки проектирования и разработки архитектуры программного комплекса;
  - ошибки корректности формирования и выполнения требований к компонентам и программному комплексу;
  - ошибки проектирования и разработки модулей и компонентов комплексов программ.

Рис. 1.1

Исследования показывают, что предприятия, пытающиеся достичь определенного уровня TMM, должны обладать, по крайней мере, тем же уровнем модели CMMI. Модель TMM хорошо подходит для автоматизированного тестирования программных комплексов потому, что эффективные программы проверки и оценки проистекают из программ производства, которые спланированы, выполняются, управляются и отслеживаются надлежащим образом. Хорошая программа тестирования не существует в вакууме; она должна быть составной частью всего процесса разработки программного комплекса. Представленный ниже перечень отражает уровни зрелости, которые имеют непосредственное отношение к организованному, автоматизированному тестированию программ, в учебнике внимание сосредоточено на четвертом и пятом уровнях.

Уровень 1 TMM– начальный. Тестирование носит хаотический характер; оно плохо определено и не отличается от отладки. Тесты разрабатываются специальным образом после завершения кодирования. Тестирование и отладка сочетаются с целью устранения ошибок программ. Целью тестирования является демонстрация того, что разработанные программы работают. Программные продукты внедряются без обеспечения качества. Ресурсы, средства и надлежащим образом подготовленный персонал отсутствуют. На этом уровне не существует цель достижения зрелости.

Уровень 2 TMM – фаза определения, тестирование отделено от отладки и определено как фаза, следующая за кодированием. Это планируемая работа, однако планирование тестирования на уровне 2 может происходить непосредственно после кодирования по причинам, касающимся незрелости процесса тестирования. Задача тестирования на этом уровне – показать, что программы соответствуют своей спецификации. Определены основные приемы и методы тестирования. На этом уровне TMM, многие проблемы качества возникают из-за того, что планирование тестирования производится на позднем этапе жизненного цикла разработки программного комплекса. Кроме того, дефекты переходят в код из фазы требований и проектирования, поскольку никакие программы критического просмотра не касаются этой важной проблемы. Тестирование после кодирования, основанное на исполнении программ, все еще считается главной работой по тестированию.

Уровень 3 TMM – интеграция. Тестирование больше не является фазой, следующей за кодированием, оно интегрировано в полный жизненный цикл разработки программного комплекса. Предприятия могут использовать навыки по планированию тестирования. Уровень 3 начинается с фазы разработки требований, продолжается на протяжении всего жизненного цикла и поддерживается версией **V** – модели. Цели тестирования устанавливаются с учетом требований, основанных на нуждах пользователя, и используются для проектирования сценариев тестирования и критерия успешного прохождения тестов. Существует подразделение по тестированию, и тестирование признано профессиональной деятельностью. Тестирование находится в центре организации технического производства комплекса программ. Основные инструментальные средства поддерживают главные работы по тестированию. На этом уровне предприятие начинает осознавать важную роль критического просмотра при контроле качества, но просмотры еще не проводятся в ходе всего жизненного цикла. Программа измерения тестирования еще не применяется к атрибутам качества процессов и продукта.

Уровень 4 TMM – управление и измерение. Тестирование – измеряемый и выражаемый в количественной форме процесс. Проверки на всех фазах процесса разработки признаны работами по тестированию и контролю качества. Программные продукты тестируются по таким атрибутам качества, как корректность, надежность, удобство использования и возможность сопровождения. Тестовые сценарии из всех компонентов собираются и записываются в базу данных конфигурационного управления тестовых сценариев для повторного использования и регрессионного тестирования. Дефекты фиксируются, и им присваивается уровень серьезности. Ошибки в процессе тестирования возникают из-за отсутствия философии предотвращения дефектов, автоматизированной поддержки сбора, анализа и распространения метрик ошибок, относящихся к тестированию.

Уровень 5 TMM – оптимизация, предупреждение дефектов и управление качеством. Поскольку в результате достижения зрелости на предыдущих уровнях получена некоторая инфраструктура, теперь процесс тестирования определен и управляем, а его стоимость и эффективность могут контролироваться. Механизмы уровня 5 способствуют повышению гибкости и постоянному совершенствованию тестирования. Практикуются предотвращение дефектов и управление каче-

ством. Процесс тестирования определяется статистическими выборками и измерениями уровней доверия, достоверности и надежности компонентов и комплекса программ. Утверждается процедура выбора и оценки средств тестирования. Автоматизированные средства поддерживают выполнение и повторное использование тестовых сценариев, обеспечивают поддержку элементов, относящихся к тестированию, сбору и анализу дефектов, а также к сбору, анализу и применению метрик качества.

Модель организации процессов тестирования модулей, компонентов и комплексов программ

В учебнике в качестве основой используется традиционная **V – модель** организации, взаимодействия процессов формирования и декомпозиции требований к функциям, характеристикам и структуре комплексов программ и последовательного комплексирования и тестирования для удовлетворения этих требований – рис. 1.2. Анализ, проектирование и производство отражаются ***сверху вниз*** левой стороной **V** – модели, а сборка, тестирование и испытания ***снизу вверх*** – правой стороной схемы. Эта модель иллюстрирует естественные потоки и взаимодействие процессов разработки требований и тестирования при последовательной реализации основных базовых элементов сложных программных комплексов, компонентов и модулей. При этом важную роль играет организация коллектива специалистов, реализующих процессы тестирования.

Основой процессов организации тестирования являются ***требования к функциям и характеристикам системы*** управления и обработки информации, использующей комплекс программ. ***Системные, функциональные требования*** должны отражать потребности к задачам и функциям программного комплекса, содержащего взаимосвязанные компоненты и модули. При этом система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть оперативный персонал, выполняющий определенные функции системы. Программные комплексы все больше встраиваются в различные системы. Работа над такими проектами требует от коллектива квалифицированных программных специалистов широкого взгляда на общие ***задачи проектирования систем***.

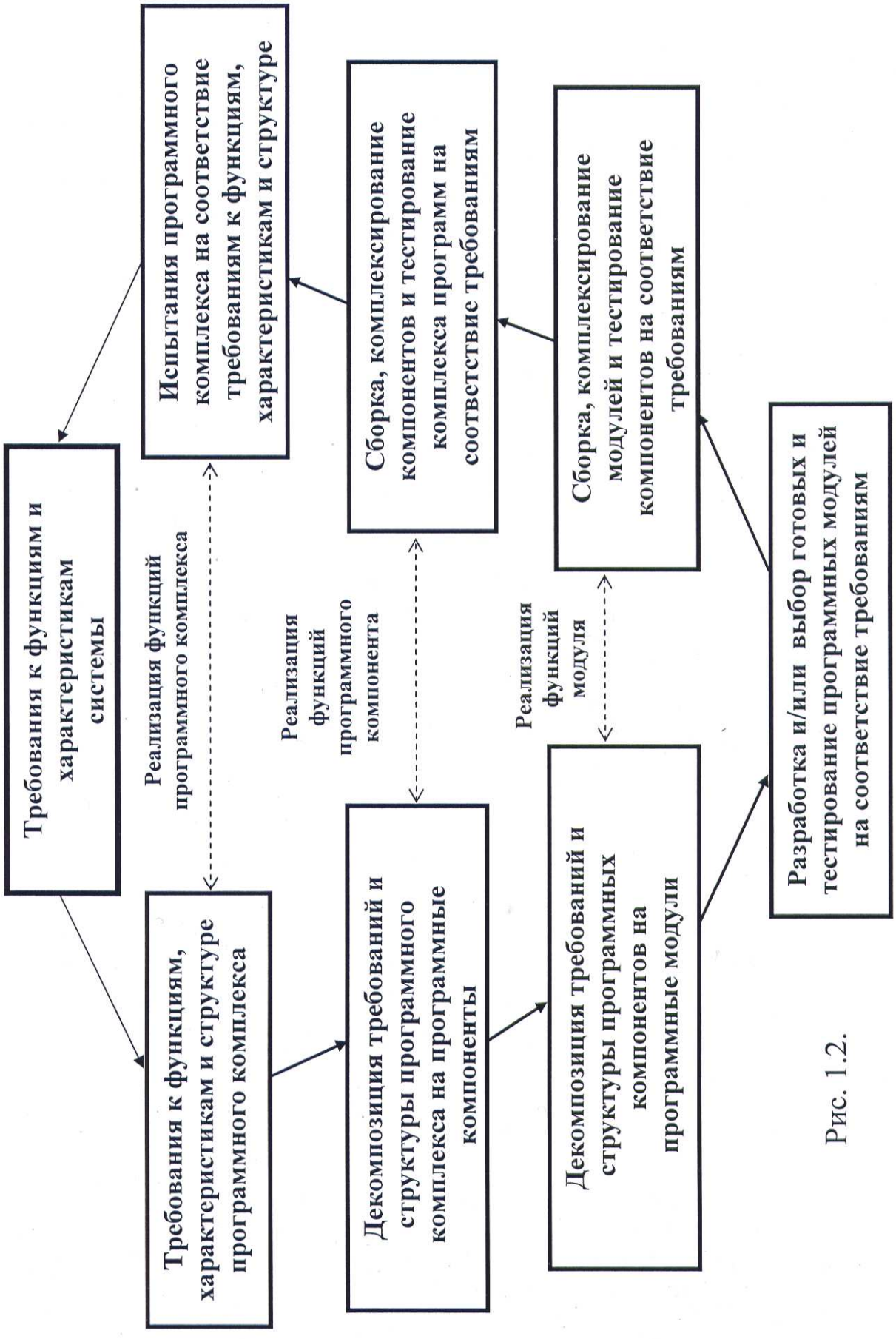

Требования к функциям и характеристикам системы
Испытания программного комплекса на соответствие требованиям к функциям, характеристикам и структуре
Реализация функций программного комплекса
Требования к функциям, характеристикам и структуре программного комплекса
Сборка, комплексирование компонентов и тестирование комплекса программ на соответствие требованиям
Реализация функций программного компонента
Декомпозиция требований и структуры программного комплекса на программные компоненты
Сборка, комплексирование модулей и тестирование компонентов на соответствие требованиям
Реализация функций модуля
Декомпозиция требований и структуры программных компонентов на программные модули
Разработка и/или выбор готовых и тестирование программных модулей на соответствие требованиям

Рис. 1.2.

Программным менеджерам и аналитикам необходимо участвовать в выработке требований для всей системы, а также освоить прикладную область, требованиям которой должен будет отвечать программный продукт.

Спецификации требований к системе являются результатом системных решений и источником для соглашений о создании или приобретении функциональных программных компонентов и критерия их приемки. Они также являются основой для принятия решений по производству или приобретению и повторному использованию компонентов для их верификации и для установления стратегии комплексирования этих компонентов в сложный комплекс программ. Порядок проектирования архитектуры должен позволять проводить анализ изменений требований в течение жизненного цикла системы, а также является источником информации для последующего повторного использования архитектуры и компонентов. Также это источник информации, при помощи которой ***определяются необходимые тесты*** в ходе комплексирования компонентов системы, которые ниже представлены процессами V – модели с краткими комментариями (см. рис. 1.2).

Требования к функциям, характеристикам и структуре программного комплекса формируются на основе заданных функций, свойств и характеристик системы и должны удовлетворить возможность его применения по основному назначению с требуемым качеством. Они должны решать весь комплекс задач системы с учетом ограниченных ресурсов.

Декомпозиция требований и структуры программного комплекса на программные компоненты необходима для снижения сложности и реструктуризации компонентов комплекса, позволяющей вести разработку и тестирование программ относительно небольшими группами специалистов, которым представляются корректные требования к функциям, характеристикам и качеству определенных компонентов.

Декомпозиция требований и структуры программных компонентов на программные модули должна обеспечивать возможность корректной формализации требований к функциям и характеристикам каждого модуля, тестируемого определенным тестировщиком, гарантирующим его качество и возможность применения в определенных программных компонентах.

Разработка и/или выбор готовых и тестирование программных модулей на соответствие с требованиям должны гарантировать их качество и обеспечивать возможности включения в программные компоненты. Модули являются наиболее массовыми элементами комплексов программ, в их создании участвуют наименее квалифицированные специалисты и при тестировании выявляется наибольшее количество дефектов и ошибок.

Сборка, комплексирование модулей и тестирование компонентов на соответствие требованиям должны сформировать и подтвердить реализацию функций достаточно крупными функциональными компонентами высокого качества, готовыми к встраиванию в комплекс программ.

Сборка, комплексирование компонентов и тестирование комплекса программ на соответствие требованиям должны завершить тестирование комплекса с полным набором компонентов, соответствующих требованиям, и реализацию всех функций программного комплекса.

Испытания программного комплекса на соответствие требованиям к функциям, характеристикам и структуре должны обеспечить корректное решение и контроль всех функциональных задач в соответствие с требованиями к системе, пригодность для поставки и внедрению в систему заказчика и пользователям.

Организация и руководители коллектива специалистов для тестирования компонентов и программных комплексов

Коллективам – командам тестировщиков должны быть присущи ***свойства всех производственных коллективов***. Создание коллектива специалистов представляет собой процесс планируемого и хорошо продуманного стимулирования к эффективному труду при одновременном сведении к минимуму трудностей и препятствий, мешающих проявлению навыков и изобретательности членов коллектива. На ***эффективность результатов коллективной работы*** влияют следующие основные факторы:

- команда должна иметь правильное соотношение навыков, опыта и личностных качеств членов группы;
- члены группы должны воспринимать себя как единую ***ко-***

манду, а не как простую совокупность индивидуумов, работающих над одной проблемой;

- между членами команды должны быть дружеские отношения;
- необходимо организовать команду таким образом, чтобы каждый чувствовал свою ценность и был удовлетворен своей ролью в результатах.

Организация команды, которая могла бы эффективно работать над комплексом программ, является сложной задачей для руководителя. Необходимо, чтобы в команде было ***равное соотношение технических навыков, опыта и выражения индивидуальности***. Наилучший способ воспитать дух команды – дать возможность каждому почувствовать, что он несет ***определенную долю ответственности за результаты*** и что ему доверяют, а также гарантировать доступ к проектной информации для всех членов коллектива. Регулярный обмен информацией – самый дешевый и эффективный способ дать людям почувствовать себя частью команды.

Формирование команды, которая могла бы эффективно работать над компонентом, функцией или комплексом программ, является достаточно сложной задачей. Необходимо обучать подчиненных справляться с большим напряжением и поощрять ***профессиональную работу во взаимодействии и по графику***. Для обеспечения приверженности подчиненных выполняемой работе следует нанимать людей, цели которых совпадают с целями предприятия, а также людей, высоко ценящих конкретную работу в целом. Полезно как можно раньше поставить перед сотрудниками интересные задачи и обеспечить условия для их удовлетворенности результатами собственного труда.

Для создания ***благоприятной, здоровой и безопасной рабочей обстановки*** в коллективе целесообразно давать подчиненным разнообразные, интересные задачи, обеспечить социальные гарантии, подбирать сотрудников, с которыми приятно работать, добиваться положительных откликов заказчиков, заручиться доверием сотрудников, уделять внимание к работе других рабочих групп. Необходимо также вовлекать сотрудников в процесс принятия решений и поощрять хорошую работу продвижением по службе и предоставлением особых прав. Для эффективного выполнения проектных работ следует предоставить все необходимые ресурсы и обеспечить безопасность работ. Для поддержания производительности труда на должном уровне

полезно разработать программу оценки и контроля сроков и качества выполненных работ, а также обучения персонала. Следует создать у подчиненных уверенность в том, что администрация готова решать их проблемы, справедливо распределять премии, выдвигать сотрудников только по их деловым качествам, продвигать новые идеи, сохранять лояльность к работникам, предоставлять возможности для роста кадров, а также доступ к информации.

Большую роль в деле ***стабилизации*** коллектива играет ***социально-психологический климат.*** Это психологическое состояние коллектива, характер ценностных ориентаций, межличностных отношений и взаимных ожиданий в нем. Оно зависит от среды и уровня развития коллектива и непосредственно влияет на эффективность деятельности его членов, на осуществление основных его функций. Благоприятным является климат такого коллектива, ценности и отношения в котором отвечают ***задачам развития коллектива***: у членов достаточно развита потребность в труде как сфере актуализации личности; в межличностных отношениях развиты взаимное доверие и уважение друг к другу, взаимная информированность по значимым вопросам, взаимовыручка и взаимная ответственность.

Основная цель квалифицированных руководителей тестирования – обеспечить функционирование программного продукта в ***соответствии с установленными и утвержденными требованиями*** при любых заданных условиях и данных. Комплекс должен удовлетворять требованиям тест-менеджера и конечных пользователей, при этом должно быть выявлено и устранено как можно больше его дефектов. Кроме того, тестирование должно обеспечивать характеристики качества, достаточные для принятия решения о готовности программного компонента для комплексирования с другими компонентами системы и его применения по назначению в целом.

Существенным для сложных проектов является ***участие профессиональных руководителей по тестированию в разработке требований к программным компонентам и комплексу***. На этапе определения требований руководители тестирования должны способствовать созданию ясных и непротиворечивых требований, рассматривать конфликтующие пожелания, поступающие от различных участников проекта комплекса программ и находить компромиссы, необходимые для определения приоритетов набора функций, представляющих наибольшую ценность тестирования для максимального чис-

ла участников проекта, прежде всего, пользователей. Руководители должны вести переговоры с заказчиком, руководством системы, пользователями и разработчиками и поддерживать ***равновесие*** между тем, чего хочет руководитель производства комплекса программ, и тем, что может предоставить команда разработчиков программ и тестировщиков за ресурсы и время, отведенные для их реализации. Участие тестировщиков на этом этапе нужно еще и для того, чтобы обеспечить формулировку свойств совокупности требований в пригодных для тестирования терминах. При определенном исходном состоянии системы и множестве входных параметров тестировщики должны иметь возможность предсказать состав и содержание выходных ***эталонных данных*** тестирования комплекса программ.

Руководство тестированием компонентов и комплекса программ могут осуществлять один или два лидера – менеджера с различными функциями. ***Менеджер проекта программного продукта*** – этот специалист обеспечивает коммуникацию между заказчиком и проектной командой, его задача – определить и обеспечить удовлетворение требований заказчика с учетом доступных ресурсов. Менеджер проекта является высшим должностным лицом, принимающим важнейшие решения по внесению изменений и корректировке требований и конфигурации сложных комплексов программ. Он взаимодействует с заказчиком и пользователями, определяющими модификации, для согласования изменений требований к системе. Заказчик системы должен оценивать и утверждать наиболее крупные изменения, заметно влияющие на условия контракта, технические требования или стоимость программного продукта.

Менеджер – системный архитектор программного продукта – управляет коммуникациями и взаимоотношениями в проектном коллективе, является координатором тестирования компонентов и комплекса программ, разрабатывает базовые, функциональные спецификации требований и управляет ими, ведет график управления тестированием проекта и отчитывается за его состояние, инициирует принятие критичных для хода проекта модификаций. Он должен организовывать структуру и состав коллектива квалифицированных специалистов для тестирования, выделять и учитывать личные свойства и психологические характеристики специалистов при организации «команд» для эффективного тестирования модулей, компонентов и комплексов программ.

В стратегии тестирования следует учитывать характеристики системы: количество компонентов программного продукта, типы, размер, критичность и безопасность создаваемых и применяемых компонентов и документов. Необходимо осуществлять проверку спецификаций требований к компонентам программного комплекса, чтобы удостовериться, что они соответствуют базовой концепции проекта и функций программного продукта, осуществлять управление изменением приоритетов тестирования задач, функций и компонентов, а также добавлением или исключением новых функций компонентов комплекса программ.

Руководители, контролирующие и управляющие обеспечением качества программных продуктов, должны овладеть стандартами и методиками предприятия, поддерживающими регистрацию, контроль, документирование и воздействия на показатели качества на этапах тестирования комплексов программ. Они должны обеспечивать эксплуатацию системы качества проекта, выявление всех отклонений от заданных показателей качества объектов и процессов, а также от предписанной технологии на этапах сопровождения и управления конфигурацией. Осуществлять выбор, организацию и упорядочение рентабельных стратегий, процессов и методов тестирования компонентов и комплексов программ, анализировать и выбирать приоритеты сценариев и значений тестов для эффективного тестирования компонентов и комплексов программ. Эти же специалисты должны анализировать возможные последствия выявленных отклонений от требований технического задания или спецификации на изменения. В результате должны приниматься меры либо по устранению отклонений, либо по ***корректировке требований***, если устранение отклонений требует чрезвычайно больших ресурсов.

Разработке тестовых процедур должны предшествовать работы руководителей по ***настройке и установке тестовой среды, по подготовке документации необходимой для регламентирования тестирования***. Руководитель тестирования должен обеспечивать инструкции по разработке тестовых процедур и применению генераторов динамических тестов, которые будут использоваться тестировщиками. Сложность необходимых тестов для контроля функции программного комплекса, обычно такая же, как сложность ***разработки*** и ***программирования*** соответствующей функции. Покрытие их тестами для устранения дефектов и обеспечения качества – это последова-

тельное покрытие модулей, компонентов или комплексов программ из ***пространства применяющихся тестов*** для их контроля, которое должно соответствовать ***пространству требований*** к функциям и характеристикам программного продукта.

Прогнозирование руководителями затрат ресурсов на тестирование компонентов и комплексов программ, возможно более или менее корректно, на основе обобщения статистических данных ряда предшествующих проектов. Обычно наиболее важным для реализации проекта и зависящим от большинства его особенностей и факторов является ***трудоемкость, непосредственно определяющая стоимость и длительность*** тестирования создаваемого комплекса программ и его компонентов. Следует оценивать рентабельные, допустимые сроки и ресурсы для завершения плана тестирования программного комплекса. Ограничения реальных ресурсов на верификацию и тестирование определяют ***достижимое качество*** версий программных продуктов.

Анализ и мониторинг характеристик, последствий и частости появления при тестировании выявленных дефектов конкретных модулей программы может служить ***ориентиром руководителям для оценки индивидуальной профессиональной квалификации*** и качества работы определенных ***программистов***, разработчиков компонентов и комплексов программ. Следствием такого анализа может быть выделение некоторых специалистов – разработчиков программ, отличающихся большим числом обнаруживаемых тестировщиками дефектов, для их замены или дополнительного обучения с целью сокращения в компонентах числа первичных ошибок соответствующего типа. Накопление, классификация и обобщение характеристик дефектов определенных классов позволяет прогнозировать при тестировании обнаружение ошибок определенных программистов, а также ***необходимое распределение*** состава и квалификации специалистов тестировщиков по компонентам и функциям в составе комплекса программ.

Организационная структура коллектива специалистов при производстве сложных комплексов программ должна учитывать: цели и функции тестирования; взаимодействующие организации; службы проектирования; систему обеспечения качества и средства, которые могут быть привлечены, учитывать, если необходимо, субподрядчиков и поставщиков. В процессах разработки и тестирования

сложного программного комплекса может участвовать большое число специалистов различных направлений и квалификации, которых целесообразно объединять в единый коллектив – ***службу «команду» тестирования на соответствие требованиям и управления конфигурацией программного продукта***. Организационная структура тестирования представляет наибольшее значение с точки зрения постоянного повышения ***зрелости*** процессов и возможностей тестирования и может включать следующие основные ***службы и группы***.

Служба тестирования – поиск ошибок и недостатков программы, их описание и предоставление этой информации всем, кому она необходима. Решений относительно выпуска продукта руководитель службы тестирования не принимает, он только предоставляет руководству информацию о том, насколько продукт протестирован и каково его качество. В некоторых предприятиях основными тестировщиками считаются сами программисты, а служба тестирования им только помогает, она отвечает за техническую сторону этой работы: анализ объекта тестирования, проектирование и подготовку тестов, их выполнение и документирование. Все это требует определенной профессиональной квалификации.

Как известно, многим людям свойственно избегать ответственности и по возможности перекладывать ее на других. Именно поэтому руководители проекта часто пытаются переложить ответственность за качество продукта на службу тестирования. За качество продукта отвечает руководитель проекта. Служба тестирования только снабжает его технической информацией, сопровождая данные собственной интерпретацией. Все это вовсе не означает, что служба тестирования вообще ни за что не отвечает. Она отвечает за качественное тестирование, интерпретацию его результатов и их своевременное предоставление руководству, документирование своей работы. Участие службы тестирования в управлении проектом скорее косвенное, чем непосредственное. Ее сила заключается в собираемых данных и умении правильно их представить.

Служба поддержки разработки является расширением концепции службы тестирования. Обе они являются службами, а значит, предоставляют чисто технические услуги – это не административные, не контролирующие и, как правило, аполитичные группы. Они помогают улучшить продукт, созданный другими сотрудниками (программистами), используя для этого профессиональные навыки, которых у

программистов нет. Если служба тестирования только тестирует продукт, то у службы поддержки разработки есть и другие задачи. Ее сотрудники принимают в разработке гораздо большее участие, а значит, имеют и больше возможностей для профессионального роста. Основной задачей службы поддержки разработки остается тестирование, но в зависимости от нужд конкретного предприятия могут выполняться следующие задачи. Отладка, техническая поддержка пользователей, особенно после выпуска продукта, редактирование документов руководства пользователей, анализ эксплуатационных характеристик продукта, изучение откликов пользователей.

Группа обеспечения качества – она обеспечивает качество продукта. Для этого она участвует в разработке от первого до последнего дня, устанавливая стандарты, определяя процедуры контроля и обучая людей тому, как лучше проектировать и разрабатывать программные комплексы и компоненты. Таким образом, недостатки программ не просто устраняются, а предотвращаются. Чтобы справиться со своей задачей, группа обеспечения качества должна обладать большими полномочиями, а ее сотрудники – высочайшей квалификацией в целом ряде профессий. Они должны быть высококлассными программистами, техническими писателями, руководителями, проектировщиками и аналитиками. В любом предприятии должна быть группа, отвечающая за определение стандартов, обучение персонала, управление работой и повышение ее эффективности. Именно она обеспечивает качество выпускаемых продуктов. С политической точки зрения создание в компании отдельной группы обеспечения качества – это палка о двух концах. Ведь за качество продукта должен отвечать каждый, кто, так или иначе, участвует в разработке, и особенно руководство предприятия. Если же у людей появляется хоть малейшая возможность переложить эту ответственность на кого-то другого, они немедленно ею пользуются.

Группа контроля качества – это очень влиятельное подразделение. Инспектор группы контроля качества может задержать выпуск продукта до тех пор, пока не будут соблюдены все стандарты, процедуры тестирования и исправлены все ошибки. Руководство предприятия реагирует на такие события немедленно и может запросто отменить решение группы контроля качества и распорядиться выпускать продукт, каким бы ни было его качество. Группа тестирования помогает руководству предприятия, предоставляя информацию о текущих

проблемах разработки и степени их серьезности. Однако предоставление информации это одно, а принятие решений – совсем другое. Здесь группа контроля качества обладает несколько более высокими полномочиями, чем обычная группа тестирования, поскольку может задержать выпуск продукта, не удовлетворяющего определенным требованиям.

При производстве сложных комплексов программ большими коллективами значительно повышается роль ***квалификации лидеров – руководителей*** проекта, что непосредственно отражается на производительности и результатах труда всего коллектива. Известны случаи, когда вследствие квалификации руководителей сложных программных комплексов, суммарные затраты на разработку изменялись в несколько раз как в лучшую, так и в худшую сторону. Некоторые методы учета характеристик лидеров, организации, структуры коллектива и процессов производства, позволяют ***сокращать негативное влияние человеческого фактора***.

Во всякой иерархии каждый сотрудник имеет тенденцию достигать ***своего уровня некомпетентности***. Если человек успешно справляется со своими обязанностями, его считают подходящей кандидатурой для повышения статуса. После ряда выдвижений он достигает уровня, где обнаруживается его некомпетентность, так как его новые обязанности оказываются ему не по силам. Больше его не повышают, но он остается на том месте, куда попал, хотя с обязанностями своими по-прежнему справиться не в состоянии. Каждая разновидность способностей по-своему вполне реальна, но трудно сравнима с компетентностью, предполагаемой у претендентов на руководящую должность. Этот процесс приводит к тому, что многие, особенно ***руководящие, должности заняты некомпетентными людьми***, долго остающимися на своих постах. Целесообразно отказываться от всякого повышения в должности, пока специалист еще находится на уровне своей компетентности. Большинство повышений определяются компетентностью кандидата, обнаруженной им на низшей ступени деловой лестницы.

Установление источников и типов дефектов и ошибок в компонентах и сложных комплексах программ

Для эффективной организации процесса тестирования руководителям и специалистам полезно знать основные источники и типы дефектов и ошибок, которые допускаются на начальных этапах проектирования сложных комплексов программ и комментируются ниже. ***Понятие дефекта или ошибки в программе***, в общем случае, подразумевает неправильность, погрешность или неумышленное искажение объекта или процесса, что может быть ***причиной ущерба – риска*** при функционировании и применении программного продукта. При этом должно быть ***известно или задано требование или правильное, эталонное состояние объекта или процесса***, по отношению к которому может быть определено наличие отклонения – ошибка или дефект (см. лекцию 1.2). Исходным эталоном обычно является спецификация требований заказчика или потенциального пользователя, предъявляемая к программному компоненту или комплексу. Подобный документ устанавливает состав, содержание и значения результатов применения программы, которые должен получать тестировщик или пользователь при определенных условиях и исходных данных. Любое отклонение результатов функционирования программы от предъявляемых к ней требований и сформированных по ним эталонов-тестов, следует квалифицировать как ***ошибку – дефект в программе***, наносящий некоторый ущерб при ее применении. Различие между ожидаемыми и полученными результатами функционирования комплекса программ могут быть следствием ошибок не только в созданных программах, но и ошибок в первичных требованиях спецификаций, явившихся базой при создании эталонов. Тем самым проявляется объективная реальность, заключающаяся в невозможности абсолютной корректности и полноты исходных требований и эталонов для программных компонентов и комплексов.

Источниками ошибок в комплексах программ являются специалисты – конкретные люди с их индивидуальными особенностями, квалификацией, талантом и опытом. При этом можно выделить предсказуемые дефекты требований, расширения или совершенствования компонентов и комплексов программ, и необходимые изменения, обусловливающие выявление ***случайных, непредсказуемых дефектов и ошибок***. Вследствие этого плотность потоков и размеры необ-

ходимых корректировок в требованиях к комплексу и компонентам программ могут различаться в десятки раз. Однако в сложных комплексах программ статистика и распределение типов ошибок и выполняемых изменений для коллективов разных специалистов нивелируются и проявляются достаточно общие закономерности, которые могут ***использоваться как ориентиры*** при их выявлении. Каждому типу необходимых корректировок соответствует более или менее определенная категория специалистов, являющихся источником дефектов данного типа (таблица 1). Такую корреляцию целесообразно учитывать как ***общую качественную тенденцию*** при анализе и поиске их причин ошибок. Этому могут помогать оценки типовых дефектов и корректировок, путем их накопления и обобщения по опыту разработки определенных классов комплексов программ в конкретных предприятиях.

В процессе жизненного цикла комплекса программ его требования подвергаются декомпозиции на спецификации модулей, программных и информационных компонентов. Эти спецификации рассматриваются как частные эталоны для составных частей комплекса, однако они редко бывают абсолютно полными и корректными. В процессе декомпозиции и верификации исходных требований возможно появление ошибок в спецификациях на группы программ и на отдельные компоненты. Это способствует расширению спектра возможных дефектов и вызывает необходимость создания ***гаммы методов и средств тестирования*** для выявления некорректностей в спецификациях и компонентах разных уровней детализации комплексов программ.

Статистика ошибок и дефектов в компонентах и комплексах программ и их характеристики в конкретных типах проектов могут служить ***ориентирами*** для разработчиков при распределении ресурсов в жизненном цикле комплексов программ и предохранять их от излишнего оптимизма при оценке достигнутого качества программных комплексов. Характеристики дефектов и ошибок представлены в ряде монографий, посвященных тестированию и экономике производства комплексов программ. Эти публикации целесообразно изучать и использовать как ориентиры в реальных проектах (см. рис. 1.1).

Таблица 1

Типы изменений требований, право на их выполнение и утверждение

Объекты изменения требований	**Типы изменений требований объектов**	**Право на выполнение изменения требований имеют:**	**Право на утверждение изменения требований имеют:**
Требования к версиям модулей и компонентов программ и документов	Устранение дефектов в требованиях к программным модулям и компонентам	Программисты – разработчики требований к модулям и компонентам	Руководители разработки требований к функциональным группам программ
Требования к версиям функциональных компонентов программ и документов	Корректировки требований к функциям и взаимодействию программных компонентов	Руководители разработки требований к функциональным компонентам	Менеджер-архитектор требований к программному продукту
Требования к версии программного продукта и комплекту документации	Требования к модификации и улучшению функций и качества версии программного продукта	Менеджер - архитектор требований к программному продукту	Менеджер и заказчик проекта программного продукта
Версии требований пользователей к программному продукту	Адаптация требований к характеристикам внешней среды пользователей	Заказчик или сопровождающий версию программного продукта пользователей	Менеджер, сопровождающий версию программного продукта пользователей

Изучение и прогнозирование характеристик дефектов и ошибок в программах непосредственно связано с достигаемой корректностью, безопасностью и надежностью функционирования комплексов программ и ***помогает***:

- оценивать реальное состояние проекта и планировать необходимую трудоемкость и длительность для его завершения и устранения доступных ошибок;
- выбирать методы и средства автоматизации тестирования компонентов комплекса программ, адекватные текущему состоянию производства, наиболее эффективные для устранения определенных видов дефектов;
- рассчитывать необходимую эффективность контрмер и дополнительных средств оперативной защиты от потенциальных дефектов и не выявленных ошибок;
- оценивать требующиеся ресурсы, с учетом затрат на реализацию контрмер при модификации программ для устранения дефектов и ошибок.

На практике исходные требования – эталоны поэтапно уточняются, модифицируются, расширяются и детализируются по согласованию между заказчиком и разработчиками. Базой таких уточнений являются ***неформализованные представления и знания*** специалистов-заказчиков и разработчиков, а также результаты промежуточных этапов проектирования и тестирования. Однако установить не корректность таких эталонов еще труднее, чем обнаружить дефекты в программах, так как принципиально отсутствуют точные, формализованные данные, которые можно использовать как исходные. В процессе декомпозиции и верификации исходной спецификации требований, возможно появление ошибок в спецификациях на компоненты программ и на отдельные модули. Это способствует расширению спектра возможных дефектов и вызывает необходимость создания гаммы методов и средств тестирования для выявления некорректностей в спецификациях на компоненты разных уровней.

Важной особенностью процесса выявления ошибок в программах обычно является ***отсутствие полностью определенной программы-эталона***, которой должны соответствовать текст и результаты функционирования разрабатываемой программы. Поэтому установить наличие и локализовать дефект непосредственным сравнением с программой без ошибок в большинстве случаев невозможно. При

тестировании обычно сначала обнаруживаются ***вторичные*** ошибки и ***риски***, т.е. последствия и результаты проявления некоторых внутренних дефектов или некорректностей программ (рис. 1.3).

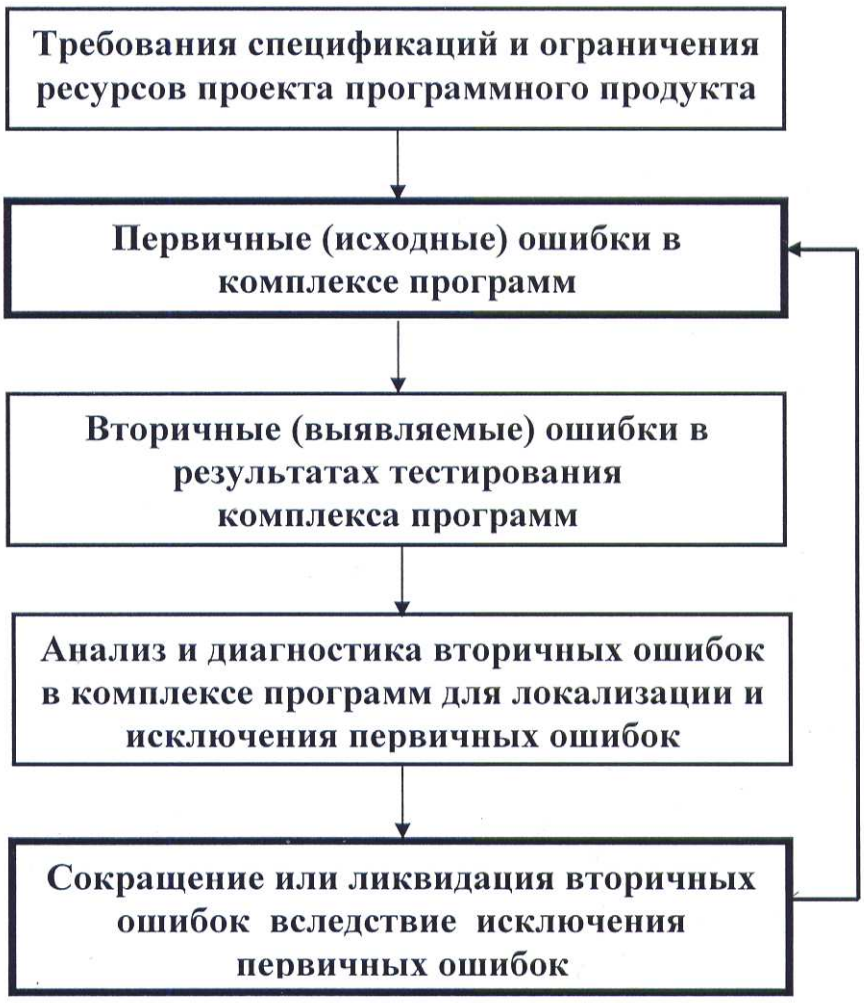


Рис. 1.3

Эти внутренние ***дефекты*** следует квалифицировать как ***первичные*** ошибки или причины обнаруженных аномалий результатов. Последующая локализация и корректировка таких первичных ошибок должна приводить к устранению ошибок, первоначально обнаруживаемых в результатах функционирования программ.

Потери эффективности и риски программ за счет неполной корректности в первом приближении можно считать прямо пропорциональными (с коэффициентом) вторичным ошибкам в выходных результатах. Типичным является случай, когда одинаковые по величине и виду вторичные ошибки в различных результирующих данных существенно различаются по своему воздействию на общую эффективность и риски применения комплекса программ. Это влияние вторич-

ных ошибок, в лучшем случае, можно оценивать методами экспертного анализа при условии предварительной, четкой классификации видов возможных первичных ошибок в программах и выходных величин. Таким образом, оценка последствий, отражающихся на вторичных ошибках и функционировании программ, может, в принципе, производиться ***по значениям ущерба – риска вследствие не устраненных их причин – первичных ошибок в программе***. Вторичные ошибки являются определяющими для эффективности функционирования программ, однако не каждая первичная ошибка вносит заметный вклад в выходные результаты. Вследствие этого ряд первичных ошибок может оставаться не обнаруженным и, по существу, не влияет на функциональные характеристики компонента или комплекса программ.

Ошибкам в программах, естественно, соответствует их обнаружение и устранение на ***основе вторичных проявлений***. Наибольшее число первичных ошибок вносится на этапах системного анализа, программирования, разработки или модификаций текстов программ. При этом на долю системного анализа приходятся наиболее сложные для обнаружения и устранения дефекты. Общие тенденции состоят в быстром росте затрат на выполнение каждого устранения ошибки на последовательных этапах процессов разработки компонентов и комплекса программ. При системном анализе интенсивность обнаружения ошибок относительно не велика и ее трудно выделить из процесса проектирования компонента или комплекса программ. Интенсивность проявления и обнаружения вторичных ошибок наиболее велика на этапе активного тестирования и автономной отладки программных компонентов. Затем она снижается приблизительно экспоненциально. Различия интенсивностей ***устранения первичных ошибок на основе их вторичных проявлений*** и внесения первичных ошибок при корректировках программ определяют скорость достижения заданного качества компонентов и комплексов программ.

Первичные ошибки в программах можно анализировать с разной степенью детализации и в зависимости от различных факторов. Практический опыт показал, что ***наиболее существенными факторами, влияющими на характеристики обнаруживаемых ошибок, являются***:

- методология, технология и уровень автоматизации системного и структурного проектирования компонентов и комплекса программ, а также непосредственного программирования компонентов;
- длительность с начала процесса тестирования компонентов и комплекса и текущий этап разработки или сопровождения комплекса программ;
- класс комплекса программ, масштаб (размер) и типы компонентов, в которых обнаруживаются ошибки;
- методы, виды и уровень автоматизации верификации и тестирования, их адекватность характеристикам компонентов и потенциально возможным в программах ошибкам;
- виды и достоверность эталонов-тестов, которые используются для обнаружения ошибок.

Одной из ***основных причин ошибок*** в сложных комплексах программ являются ***организационные дефекты требований и эталонов к программному продукту***, которые отличаются от остальных типов и могут быть выделены как самостоятельные. Ошибки и дефекты данного типа появляются из-за недостаточного понимания руководителями и коллективом специалистов целей и функций комплекса программ, а также вследствие отсутствия четкой его организации и поэтапного контроля требований качества компонентов и продуктов. Это порождается пренебрежением руководителей к организации всего технологического процесса формализации требований сложных программных продуктов и приводит к серьезной недооценке их дефектов, а также трудоемкости и сложности их выявления. При отсутствии планомерной и методичной разработки и тестирования требований и эталонов может оставаться не выявленным значительное количество ошибок и, прежде всего, дефекты требований к взаимодействию отдельных функциональных компонентов между собой и с внешней средой. Для сокращения этого типа массовых ошибок активную роль должны играть лидеры – менеджеры и аналитики-системотехники, способные вести контроль и конфигурационное управление требованиями, изменениями и развитием версий и компонентов комплексов программ.

Системные ошибки и недостатки определения требований к программному продукту характеризуются, прежде всего, неполной информацией о реальных процессах, происходящих в источниках и потребителях информации. Кроме того, эти процессы зачастую зави-

сят от самих алгоритмов и поэтому не могут быть достаточно определены и описаны заранее без исследования функционирования комплекса программ во взаимодействии с внешней средой. На начальных этапах разработки не всегда удается точно и полно сформулировать целевую задачу всей системы, а также целевые задачи основных функциональных групп программ, и эти задачи уточняются в процессе проектирования. В соответствии с этим уточняются и конкретизируются спецификации на функциональные компоненты и выявляются отклонения от уточненного задания требований, которые могут квалифицироваться как системные ошибки.

Во многих случаях отсутствует полная адекватность условий получения предполагаемых и реальных характеристик внешней среды, что может являться причиной сложных и трудно обнаруживаемых ошибок. Это усугубляется тем, что часто невозможно заранее предусмотреть все разнообразие возможных внешних условий и реальных вариантов сценариев функционирования и применения версий программных продуктов. При автономной и в начале комплексной отладки версий компонентов относительная доля системных ошибок может быть невелика (около 10%), но она существенно возрастает (до 35 – 40%) на завершающих этапах комплексной отладки новых версий программного продукта. В процессе сопровождения системные ошибки обычно являются преобладающими (около 60 – 80% от всех ошибок).

Ошибки определения характеристик системы и внешней среды, принятых в процессе разработки комплекса программ за исходные, могут быть результатом аналитических расчетов, моделирования или исследования аналогичных систем. В ряде случаев может отсутствовать полная адекватность предполагаемых и реальных характеристик, что является ***причиной сложных и трудно обнаруживаемых системных ошибок и дефектов развития проекта***. Ситуация с такими ошибками дополнительно усложняется тем, что эксперименты по проверке взаимодействия программного продукта с реальной внешней средой во всей области изменения характеристик зачастую сложны и дороги, а в отдельных случаях, при создании опасных ситуаций, недопустимы. В этих случаях приходится использовать моделирование и имитацию внешней среды с заведомым упрощением ее отдельных элементов и характеристик, хотя степень упрощения не всегда можно оценить с необходимой точностью. Однако

полной адекватности моделей внешней среды и реальной системы добиться трудно, а во многих случаях и невозможно, что может являться причиной значительного числа дефектов.

Дефекты и ошибки программных комплексов, которые проявляются ***как риски*** – прямой ущерб при применении системы. Основными источниками рисковых ситуаций могут быть некорректные исходные требования, дефекты или ошибки в программах и данных функциональных задач, проявляющиеся при их исполнении в соответствии с назначением. При таких воздействиях функциональная работоспособность систем может разрушаться не полностью, однако невозможно полноценное выполнение заданных функций и требований к качеству информации для потребителей. Риски могут быть обусловлены нарушениями и дефектами технологий или ограничениями при использовании ресурсов – бюджета, планов, коллектива специалистов, инструментальных средств, выделенных на разработку компонентов и комплекса программ. Результирующий ***ущерб*** в совокупности зависит от величины и вероятности проявления каждого дефекта и негативного воздействия. Одним из косвенных методов определения ***величины риска*** может быть оценка совокупных затрат, необходимых для ликвидации негативных последствий дефектов и ошибок в программах, системе или внешней среде, проявившихся в виде конкретного рискового события. В жизненном цикле программных комплексов ущерб – риски могут проявляться вследствие:

- искажений, дефектов или ошибок не полной реализации требуемого назначения, функций или взаимодействия комплекса программ с компонентами системы или внешней среды – недостатков и дефектов ***функциональной пригодности комплексов программ***;
- недостаточных и не соответствующих требованиям ***конструктивных характеристик*** качества компонентов и комплекса программ при его функционировании и применении по прямому назначению;
- ошибок и нарушений ограничений на использование ***экономических, временных или технических ресурсов*** при создании и применении компонентов и комплексов программ.

В зависимости от сложности проекта окончательным результатом работ при проектировании должны быть детализированные и утвержденные требования к номенклатуре, свойствам и значениям качества и ***допустимым рискам программного продукта***, которые

достаточны для его полноценного рабочего проектирования, производства и последующей эффективной эксплуатации. На этапах жизненного цикла и при конфигурационном управлении требования могут изменяться по согласованию между заказчиком и разработчиком, которые чаще всего приурочиваются к подготовке новой версии. Для этого необходим мониторинг масштаба проекта, требований и реализаций характеристик и допустимых рисков. После определения назначения и функций комплекса программ подготовка исходных данных и концепции проекта должны завершаться ***выделением номенклатуры конструктивных характеристик***, имеющих достаточно сильное влияние на функциональную пригодность и сокращающих риски до допустимых значений.

Принципиальные и технические возможности, точность реализации свойств и измерения значений характеристик комплекса, а также общие ресурсы конкретного проекта всегда ограничены в соответствии с их содержанием и возможностями заказчика и разработчиков. Это определяет рациональные диапазоны значений каждого типа допустимого риска, которые могут быть выбраны на основе требований заказчика, здравого смысла, а также путем анализа пилотных проектов и прецедентов в спецификациях требований реализованных проектов. В результате формируется полный ***набор требуемых функциональных и конструктивных характеристик, свойств и допустимых рисков комплекса программ***.

Ошибки комплексов программ по сложности обнаружения и масштабу корректировок можно разделить на следующие группы:

- ошибки, обусловленные сложностью компонентов и комплекса в целом, наиболее сильно влияющие на размеры модификаций;
- ошибки вследствие большого масштаба – размера комплекса программ, а также высоких требований к его качеству;
- ошибки планирования и корректности требований модификаций часто могут быть наиболее критичными для общего успеха комплекса программ и системы;
- ошибки проектирования, разработки структуры и функций комплекса программ в более полные и точные технические описания сценариев того, как комплекс - система будут функционировать.

Сложность обнаружения и устранения ошибок значительно конкретизируются и становятся измеримой, когда устанавливается

связь этого понятия с конкретными ресурсами, необходимыми для решения соответствующей задачи и возможными проявлениями дефектов. При разработке и сопровождении программ основным лимитирующим ресурсом обычно являются допустимые трудозатраты специалистов, а также ограничения на сроки разработки, технологию проектирования корректировок комплекса. Показатели сложности при анализе можно разделить на ***две большие группы***:

- ***сложность ошибок при создании корректировок*** компонентов и комплекса программ – статическая сложность, когда реализуются его требуемые функции, вносятся основные дефекты и ошибки;
- ***сложность проявления ошибок функционирования*** и получения результатов программных компонентов и комплекса – динамическая сложность, когда проявляются дефекты и ошибки, отражающиеся на функциональном назначении, рисках и качестве применения комплекса программ.

К группе факторов, влияющих на ***сложность ошибок*** комплексов программ, относятся:

- величина – размер создаваемой или модифицируемой программы, выраженная числом строк текста, функциональных точек или количеством программных модулей и компонентов в комплексе;
- количество обрабатываемых переменных или размер и структура памяти, используемой для размещения базы данных корректировок;
- трудоемкость разработки изменений компонентов и комплекса программ;
- длительность разработки и реализации корректировок;
- число специалистов, участвующих в производстве компонентов и комплекса программ.

Некоторые из перечисленных параметров коррелированны между собой, причем, определяющими часто являются размер изменений программы и объем базы данных. Остальные характеристики можно рассматривать как вторичные, однако они могут представлять самостоятельный интерес при анализе сложности и прогнозировании вероятного числа дефектов в измененной программе.

Масштаб – размер комплексов программ и их изменяемой

части наиболее сильно влияет на количество ошибок, а также на требования к ***качеству***. По мере увеличения размера и повышения требований к качеству комплекса программ и его корректировкам затраты на обнаружение и устранение ошибок увеличиваются все более высокими темпами. Одновременно расширяется диапазон неопределенности достигаемого качества. В зоне высокого качества программ возрастают трудности измерения этих характеристик, что может приводить к необходимости изменения затрат в несколько раз в зависимости от применяемых методов и результатов оценки качества компонентов и комплекса программ. Вследствие этого в сложных комплексах всегда велика вероятность проявления не устраненных ошибок и недостаточная достоверность оценок достигнутого качества.

Ошибки проектирования и разработки архитектуры программного комплекса определяются процессами перевода неопределенных и общих положений, сделанных на этапе первичных спецификаций требований, в более точные технические описания сценариев того, как программный продукт и система должны работать. Ошибки структуры инспекциями легче обнаружить, чем ошибки требований, но они в конечном итоге могут оказаться при корректировках такими же дорогостоящими. Главная причина того, что ошибки структуры дорого исправлять, состоит в том, что они могут влиять на систему в целом. Если разработчик структуры либо неверно прочитает требования, либо не увидит содержание требования так же, как заказчик или конечный пользователь, появится ошибка разработки структуры данного компонента или комплекса программ.

Системные ошибки корректности выполнения требований к программным компонентам считаются наиболее критичными для общего успеха его версий и системы в целом. Ошибки требований являются наиболее трудными для обнаружения и наиболее сложными для исправления. Вот почему исправление ошибок выполнения требований может быть в десятки раз дороже, чем ошибок программирования. Требование может быть пропущено в спецификации к системе, комплексу и компонентам программ. Это ведет к неудовлетворенности пользователя, и комплекс программ считается заказчиком и пользователем дефектным. Пропуск части требований – наиболее обычная проблема среди ошибок требований. Ошибки требований могут также представлять собой конфликтующие требования в спецификации. Может проявляться неопределенность требований –

такой способ формулирования требования, что даже если и не конфликтует с другим требованием, оно выражено недостаточно ясно, чтобы привести к единственному, конструктивному решению при разработке комплекса программ. Конечный пользователь часто называет это ошибкой, хотя на самом деле это выбор конструктивного решения на основе неполного или неопределенного требования. Многочисленные исследования показали, что ***ошибки требований труднее всего обнаруживать и дороже всего исправлять***.

Ошибки проектирования и разработки модулей и компонентов комплекса программ определяются процессами перевода зачастую неопределенных и общих положений, сделанных на стадии спецификаций требований, в более точные технические описания сценариев того, как компоненты программы должны работать, к ним относятся:

- системные ошибки в модулях и компонентах программ;
- алгоритмические ошибки модулей и программных компонентов;
- ошибки реализации спецификаций модулей и компонентов;
- программные ошибки модулей и компонентов комплекса программ;
- ошибки в эксплуатационной и технологической документации модулей, компонентов и комплекса программ.

Убывание ошибок в компонентах и комплексе программ и интенсивности их обнаружения в процессе разработки не беспредельно. После тестирования и отладки в течение некоторого времени интенсивность обнаружения дефектов при самых жестких внешних условиях испытаний снижается настолько, что коллектив, ведущий разработку и тестирование, попадает в ***зону нечувствительности к ошибкам и возможным отказам функционирования***. При такой интенсивности отказов вследствие их редкого проявления трудно прогнозировать затраты времени, необходимые для обнаружения очередной ошибки. Создается представление о полном отсутствии случайных проявлений дефектов, о невозможности и бесцельности их поиска, поэтому усилия на тестирование сокращаются, и интенсивность обнаружения ошибок еще больше снижается. Этой предельной интенсивности обнаружения отказов соответствует наработка на обнаруженную ошибку, при которой ***прекращается улучшение характеристик программного комплекса*** на этапах отладки или испытаний.

При серийном выпуске программного продукта, благодаря значительному расширению вариантов исходных данных и условий эксплуатации, возможно в течение некоторого времени возрастание суммарной (по всем экземплярам системы) интенсивности обнаружения дефектов и ошибок. Это позволяет дополнительно устранять ряд дефектов и тем самым увеличивать длительность между проявлениями ошибок в процессе эксплуатации. Для оценки этих показателей качества необходимы объективные данные о динамике выявления ошибок, а также об усилиях, затрачиваемых на их обнаружение и устранение с учетом объема, тиража и других параметров создаваемого программного продукта.

Лекция 1.2

ЭТАЛОНЫ И ТРЕБОВАНИЯ ПРИ ПРОЕКТИРОВАНИИ И ПРОИЗВОДСТВЕ КОМПЛЕКСОВ И КОМПОНЕНТОВ ПРОГРАММ

Системные основы разработки требований к сложным комплексам программ

Специалисты по созданию программного продукта ***должны понимать функции, задачи и методы создания сложных систем***, поскольку возникающие проблемы часто являются результатом решений, принятых руководителями и разработчиками проекта всей системы. Технологии программной инженерии часто являются критическим фактором при разработке сложных вычислительных систем. В этом состоит сложность прогнозирования и оценки их свойств, поскольку иногда можно измерить характеристики только подсистем, из которых состоит комплексная система. Высокие темпы роста основных ресурсов аппаратных средств (приблизительно на порядок каждые пять лет) и сохраняющаяся потребность в увеличении их использования со стороны различных пользователей и сфер применения приводят к необходимости ***адекватного совершенствования технологий создания сложных комплексов программ***.

Основы процессов, составляющих жизненной цикл сложных технических систем, создаваемых и применяемых человеком, отражает стандарт **ISO 15288:2002** (см. Приложение 1). Процессы в данном стандарте образуют полное множество, из которого предприятия могут конструировать модели систем, соответствующие их продукции. Стандарт может быть использован для выбора, структуризации и применения установленной среды в интересах производства продукции при оценке проекта на соответствие заявленной и сформированной внешней среде. Этот стандарт устанавливает ***основы для описания требований к функциям, процессам и свойствам жизненного цикла сложных систем***, определяет множество детально определенных процессов, требований и соответствующей терминологии. Выбран-

ные из них множества процессов могут быть использованы для управления и осуществления жизненного цикла систем. Стандарт относится к искусственным техническим системам, которые созданы или разрабатываются человеком и ***состоят из следующих компонентов***: аппаратное обеспечение, программные продукты, люди, процессы, процедуры, внешняя среда и природные ресурсы. Основные положения стандарта целесообразно ***учитывать при формировании функциональных требований к комплексам программ***, используемым в системах – рис. 1.4.

Требования к системе должны определять совокупность работ, которая позволяет в рамках предприятия и/или проекта оптимизировать прибыль и уменьшать риски, возникающие вследствие принятия проектных решений и осуществления соответствующих действий. Эти работы обеспечивают условия для того, чтобы продукция была нужной и полезной, экономически выгодной, функциональной, надежной, пригодной к обслуживанию, производству и использованию, и обладала другими качествами, необходимыми для того, чтобы удовлетворить требования как заказчика, так и пользователя. Они также обеспечивают условия для того, чтобы продукция соответствовала ожиданиям или законным требованиям общества, включая общие факторы обеспечения здоровья, безопасности, надежности и экологии.

Цель процесса определения и формирования требований заказчика состоит в формулировании требований к системе, выполнение которых должно обеспечить ***функциональные возможности***, необходимые пользователям системы и иным заинтересованным лицам, в заданной эксплуатационной среде. Должны быть определены цели создания и назначения системы, а также функции и область ее применения. Следует выявить и зафиксировать заинтересованных лиц или их группы, которые будут связаны с системой на протяжении всего жизненного цикла, а также их потребности и пожелания. Эти данные анализируются и преобразуются в общий набор требований заказчика, они описывают необходимое поведение системы в процессе взаимодействия с эксплуатационной средой и совокупность образцовых показателей, проверка на соответствие которым является целью процесса аттестации и позволяет подтвердить, что система отвечает заявленным требованиям.

Эталоны и требования при проектировании и производстве комплексов и компонентов программ должны включать:

- системные основы разработки требований к сложному комплексу программ:
  - функции, задачи и методы создания сложных систем;
  - процесс определения и формирования требований заказчика к системам;
  - определения особенностей внешней среды систем;
  - процессы проектирования архитектуры систем;
  - исходные проектные данные и требования к программному продукту;
- формализацию эталонов требований и характеристик к комплексу программ:
  - эталонный – базовый масштаб проекта комплекса программ;
  - набор функций, который необходимо реализовать, размер, сложность комплекса программ;
  - ресурсы, которые может предоставить заинтересованный заказчик или будущий пользователь комплекса программ;
  - время, которое может быть выделено на создание комплекса программ;
  - целостный комплекс эталонных требований к функциям, характеристикам, архитектуре и качеству программных компонентов и комплекса от *з*аказчика или программного менеджера проекта;
- формирование требований компонентов и модулей путем декомпозиции функций комплексов программ:
  - функциональную декомпозицию комплексов программ;
  - требования структурирования архитектуры комплексов, компонентов и модулей программ;
  - унификацию архитектуры и интерфейсов модулей, компонентов и комплексов программ;
  - общие свойства взаимодействия компонентов иерархических систем;
  - регламентирование размеров модулей и программных компонентов;
  - распределение и установление ответственности специалистов за качество и сроки создания модулей и компонентов.

Рис. 1.4

В соответствии с принятой политикой предприятия и/или проекта для ***определения требований к системе*** должны осуществляться ***следующие базовые действия***:

- идентификация заинтересованных лиц или групп, имеющих законный интерес к системе в течение ее жизненного цикла;
- определение ограничений системных решений, которые являются неизбежным следствием существующих соглашений, управленческих или технических решений;
- установление масштаба и требуемых ресурсов для реализации системы;
- определение представительного набора последовательных действий для идентификации всех требуемых функциональных возможностей, которые отвечают предполагаемым сценариям и внешним средам применения, функционирования и сопровождения системы;
- определение взаимодействия между пользователями и системой;
- анализ полноты и корректности множества установленных требований к системе;
- специфицирование экологических, медицинских, безопасности и других требований заказчика, имеющих отношение к критическим показателям применения системы;
- разрешение проблем и конфликтов, возникающих в связи с определением требований к системе;
- доведение до сведения соответствующих заинтересованных лиц результатов анализа требований для подтверждения того, что их потребности и ожидания были поняты и корректно отражены в требованиях;
- документирование требований заказчика в форме, приемлемой для управления требованиями в течение жизненного цикла системы.

В состав заинтересованных лиц могут входить: заказчики; пользователи; предприятия, занимающиеся сопровождением; разработчики; производители, поставщики и покупатели. Требования заказчика могут выражаться ***в форме потребностей, пожеланий и ограничений***. Они выражаются в виде моделей, ориентированных на цели и назначение системы и описывающих систему в контексте внешней среды и условий функционирования. Сценарии применения системы

должны использоваться для анализа ее функционирования в заданной среде с целью ***выявления требований***, которые формально могли быть не заданы заказчиком, например, юридические или социальные обязательства. Также следует анализировать социальное воздействие организации на пользователей, которые могут повлиять на использование системы или сдерживать процесс ее проектирования. Стандарты и правила должны использоваться для ***определения особенностей внешней среды системы***:

- архитектуры и ресурсов аппаратуры вычислительной системы;
- рабочих мест, внешней среды и инструментов визуализации, в том числе используемого вспомогательного оборудования;
- нормальных, необычных и чрезвычайных ситуаций функционирования системы;
- набора, обучения и развития операторов и пользователей;
- физических, умственных и научных способностей пользователей.

Следует идентифицировать угрозы безопасности и, если они имеются, то устанавливать ***требования и функции по обеспечению безопасности применения системы***. Сюда относятся риски, связанные с процессами функционирования и сопровождения, здоровьем и безопасностью, угрозами собственности и внешними воздействиями (стандарт **IEC 61508**). Необходимо устанавливать требуемые функции повышения безопасности, включая смягчение и сокращение рисков, ссылаясь на соответствующие стандарты и утвержденные профессиональные правила в случае их применимости.

Анализ корректности сформированных требований к системе и программному продукту включает выявление и идентификацию противоречивых, пропущенных, неполных, неоднозначных, нелогичных или непроверяемых требований; расстановку приоритетов и разрешение проблем, возникающих в связи с определением требований (см. рис. 1.4). Сюда же относятся требования, которые не могут быть реализованы или которые нецелесообразно реализовывать. Необходимо достигать соглашения совместно с заинтересованными лицами по решениям, касающимся противоречивых, нецелесообразных и неосуществимых требований, устанавливать, чтобы их требования были корректно откорректированы.

Цель процесса анализа требований состоит в преобразовании потребностей заказчика, выраженных в виде пользовательского представления о системе, в ***формализованные функциональные возможности***. В ходе этого процесса должно создаваться четкое представление о будущей системе, которая будет удовлетворять требованиям заказчика и не потребует специальных мероприятий в связи с ее практическим применением. В результате определяется комплекс оцениваемых требований, которые задают, с точки зрения разработчика, какими функциями и характеристиками должна обладать система и какими должны быть их значения, чтобы удовлетворить требованиям заказчика. В результате ***успешного осуществления анализа требований*** должны формализоваться:

- требуемые характеристики, свойства, функциональные и эксплуатационные требования к системе;
- ограничения, влияющие на архитектурное проектирование системы, а также на средства ее реализации;
- способы, с помощью которых обеспечиваются целостность системных требований, потребностей заказчика и взаимное соответствие между ними;
- основа для верификации комплекса системных требований.

Цель процесса проектирования архитектуры системы состоит в синтезе структурных решений, которые бы удовлетворяли системным требованиям. Они определяются на основе требований к набору системных компонентов, из которых компонуется система. Конкретные требования являются основой для верификации реализации системы и для разработки стратегий комплексирования и верификации компонентов. В результате проектирования архитектуры системы должны:

- устанавливаться порядок, в соответствии с которым выполняется проектирование архитектуры;
- задаваться реализуемый набор функциональных, системных компонентов, которые удовлетворяют требованиям, предъявляемым к системе;
- определяться требования к интерфейсам компонентов и с внешней средой при проектировании архитектуры системы;
- устанавливаться связь между проектированием архитектуры и системными требованиями;

- определяться основа для верификации требований к функциональным, системным компонентам;
- устанавливаться основа комплексирования системных компонентов разных видов.

При проектировании архитектуры следует ***определять и задавать производные требования*** для описания функциональных и эксплуатационных требований, функциональных возможностей и свойств, требований к потокам данных в соответствии с логической архитектурой. Определения требований заказчика, анализа требований и проектирования архитектуры рекурсивно применяются для последовательной детализации системной архитектуры до тех пор, пока компоненты не могут быть созданы, приобретены, повторно использованы или построены путем использования подходящего стандарта разработки, например, такого, как **ISO 12207** ***для программных компонентов***. Следующие факторы должны учитываться для достижения наиболее эффективного, экономически выгодного и надежного ***взаимодействия человека с машиной***:

- ограниченные возможности человека;
- ограничения, обусловленные действиями человека, которые могут привести к аварийной ситуации, а также ограничения, обусловленные тем, как может повлиять на ситуацию последовательность человеческих ошибок;
- интеграция человеческих возможностей в систему и ее функционирование.

Необходимо устанавливать стоимостные, технические и временные ***риски***, связанные с решениями о разработке, модификации или закупке компонентов. Следует оценивать альтернативные проектные решения, моделируя их с той степенью детализации, которая позволяет сравнивать спецификации по таким выраженным в требованиях заказчика критериям, как характеристики функционирования, стоимость, затраты времени и риски. Определения должны производиться с той степенью детализации и контроля, которая соответствует созданию, использованию и ***обеспечению целостности системы***.

Исходные проектные данные и требования к программному продукту, включая установленные законодательные и регламентирующие нормативные требования, должны быть оформлены документально, а их выбор проанализирован поставщиком на адекватность. Спецификацию требований должен представить потребитель-

заказчик. Однако по взаимному согласию ее может подготовить поставщик-разработчик в тесном сотрудничестве с потребителем для предупреждений разногласий путем, например, уточнения определений терминов, объяснения предпосылок и обоснования требований. ***Трассировка требований к системе*** должна обеспечивать связь между требованиями и отслеживание потребностей источников требований. Трассировка является фундаментальной основой проведения анализа влияния при изменении требований, помогая предсказывать эффект от внесения таких изменений. Неполные, двусмысленные или противоречивые требования должны быть предметом урегулирования с заинтересованными лицами, ответственными за их предъявление.

Для конкретного комплекса программ доминирующие требования выделяются и определяются его ***функциональным назначением***. Программы для ЭВМ как объекты проектирования, разработки, испытаний и оценки качества характеризуются в жизненном цикле следующими ***обобщенными характеристиками***:

- проблемно – ориентированной областью применения, техническим и социальным назначением программного комплекса;
- конкретным классом и назначением решаемых функциональных задач с достаточно определенной областью применения соответствующими квалифицированными пользователями;
- масштабом и сложностью комплекса программ и базы данных, решающей единую целевую задачу системы;
- архитектурой комплекса программ и базы данных;
- необходимыми составом и требуемыми значениями характеристик качества функционирования программ и величиной допустимого ущерба – риска из-за недостаточного их качества;
- составом потребителей характеристик комплекса программ, для которых важны соответствующие атрибуты качества;
- комплектом стандартов и их содержания, которые целесообразно использовать при выборе технологии и характеристик комплекса программ;
- реальными ограничениями всех видов ресурсов проекта;
- степенью связи решаемых задач с реальным масштабом времени или допустимой длительностью ожидания результатов решения задач;
- прогнозируемыми значениями длительности эксплуатации и перспективой создания множества версий программного продукта;

- предполагаемым тиражом производства и применения комплекса программ;
- степенью необходимой документированности программного продукта.

Исходные требования к комплексу программ могут быть представлены и согласованы в ***составе спецификации всей системы***. Если программный продукт нуждается во взаимодействии с другими программными или аппаратными продуктами, то в спецификации требований должны быть оговорены непосредственно или при помощи ссылок интерфейсы между разрабатываемыми и другими применяемыми продуктами. В этом случае должны быть разработаны процедуры, обеспечивающие четкое ***распределение системных требований*** между аппаратными и программными компонентами, а также соответствующие спецификации интерфейсов с внешней средой. При ***заключении контракта спецификация требований*** может быть определена не полностью, она может быть доработана в ходе реализации проекта и уточнены:

- требования к входной информации:
- источники информации и их идентификаторы;
- перечень и описание входных данных (идентификаторы, формы представления, регламент, сроки и частота поступления);
- перечень и описание структурных единиц информации выходных сообщений или ссылка на документы, содержащие эти данные;
- описание и оценка преимуществ и недостатков разработанных альтернативных вариантов функций в концепции создания проекта;
- обоснование выбора оптимального варианта требований к содержанию и приоритетам функций компонентов;
- общие требования к структуре, составу компонентов и интерфейсам с внешней средой.

Формализация эталонов требований и характеристик комплекса программ

Неформальный подход, применяющийся к построению простых программ, недостаточен для разработки больших комплексов программ. Стоимость аппаратных средств постепенно снижается, тогда как ***стоимость программных продуктов стремительно возраста-***

ет. Возникла необходимость в новых технологиях и методах управления комплексными проектами производства крупных программных продуктов. Это является одной из причин возникновения проблем при разработке сложных программных продуктов, как и то, что многие предприятия, занимающиеся их производством, не уделяют должного внимания эффективному применению современных методов и стандартов, разработанных в программной инженерии. Специалисты должны понимать, что они работают в организационных и финансовых рамках заключенных контрактов и ищут решение поставленной перед ними задачи ***с учетом условий и требований договора на систему***.

Приступая к проектированию и производству сложного комплекса программ, необходимо произвести реалистичную оценку ресурсов проекта, выделенного времени и поставленных целей. Эти факторы совместно определяют ***«эталонный – базовый масштаб» проекта*** следующими характеристиками (см. рис. 1.4):

- набором функций, который необходимо реализовать и представить для удовлетворения потребностей заказчика и пользователей комплекса программ (размер, сложность комплекса);
- ресурсами, которые может предоставить заинтересованный заказчик или будущий пользователь комплекса программ (бюджет, квалификация специалистов, качество технологии);
- временем, которое может быть выделено на создание комплекса программ.

Набор функциональных возможностей, который предполагается создать, является ***основной целью*** комплекса программ и обычно, естественно, ограничен временем и доступными ресурсами. Эти два фактора в некоторой степени взаимосвязаны, и время разработки может увеличиваться при недостаточных ресурсах. Однако не всегда увеличение ресурсов позволяет сократить время создания сложного комплекса программ. Для решения этой проблемы необходимо ***управление масштабом*** проекта комплекса программ и согласование всех изменений разработчиков и заказчиков.

Время или допустимая длительность разработки является невосполнимым ресурсом. Этот ресурс все больше определяет требования к качеству комплексов программ в процессе их разработки и сопровождения. Жесткие требования заказчиков к срокам реализации проектов естественно ограничивают разработчиков и испытателей. Увеличение числа привле-

каемых для этого специалистов только в некоторых пределах позволяет ускорять разработку. Радикальный способ увеличения реальных затрат на разработку и испытания в ограниченное время состоит в систематизации, планировании и автоматизации на всех этапах жизненного цикла комплекса программ.

Сокращение масштаба комплекса до размеров, соответствующего имеющимся времени и ресурсам, может привести к конфликтам между командой разработчиков и заказчиками, потребности которых необходимо удовлетворить. Однако можно активно привлекать заказчиков к управлению требованиями и масштабом проекта, чтобы обеспечить как качество, так и своевременность разработки программного комплекса. При этом надо учитывать, что именно заказчики несут финансовую ответственность за выполнение внешних обязательств перед клиентами-пользователями и поэтому лучшее, что может предложить команда разработчиков – это комплекс программ высокого качества, выполненный в срок и в пределах бюджета (пусть и в несколько сокращенном, в случае необходимости, масштабе). Его важнейшие функции, удовлетворяющие потребности, принадлежат заказчикам, а ***не*** команде разработчиков. Необходимы согласования с заказчиками при принятии основных решений при проектировании, и только они могут реально определить, как, сократив масштаб, получить полезный комплекс программ. Если масштаб проекта необходимо сократить, то заказчик должен являться непосредственным участником процесса уточнения требований. Таким образом, удастся избежать неприятностей с отставанием от графика и с пропущенными функциями.

Для защиты как проекта, так и целей заказчика может понадобиться вести ***переговоры*** об объеме работ для команды разработчиков. Команде следует знать, что зачастую заказчик владеет ***приемами ведения переговоров*** и, естественно, будет использовать их в дискуссии с командой. Следовательно, руководителю команды, менеджеру или лидеру проекта также необходимо овладеть соответствующими приемами. Ведение переговоров представляет собой вид профессиональной деятельности в деловой сфере. Это не особенно сложный процесс, и его можно осуществлять честно, красиво и стильно. Отделяйте человека от проблемы, концентрируйте внимание на интересах, а не позициях.

После того как базовый, ***эталонный уровень масштаба согласован и задан, он представляет собой основу***, вокруг которой концентрируется множество разнообразных видов деятельности проектирования. Функции базового уровня могут использоваться для того, чтобы реалистически оценивать прогресс в развитии проекта. Исходя из достигнутого соглашения по отношению к базовому уровню, можно манипулировать ресурсами. Базовые функции нужно разрабатывать наиболее детально, тем самым подготавливая их к разработке кода. Полезно применять трассировку потребностей пользователя к функциям эталонного уровня требований. Трассировку можно затем продолжить от функций к дополнительным спецификациям и реализации компонентов.

Наиболее важным является то, что высокоуровневый базовый уровень можно использовать ***как эталон*** для успешного управления изменениями требований. Изменения являются неотъемлемой частью любой разработки. Базовый уровень функций обеспечивает удобный механизм управления высокоуровневыми изменениями. ***Официальное изменение*** – это когда заказчик запрашивает новую возможность системы, которая не является частью базового уровня. Прежде чем включать новую функцию в базовый эталонный уровень, следует оценить воздействие этого изменения на весь проект. Если команда проекта изначально тщательно определила базовый уровень, то следует исходить из предположения, что ***любое изменение в базовом уровне повлияет на ресурсы, график или набор функций***, которые должны быть представлены в данной версии.

Функции в документе – концепции представляют собой описания желательного и полезного поведения комплекса программ на языке заказчика и пользователей. Если ресурсы фиксированы и график изменить нельзя, команда проекта должна привлечь заказчика к процессу принятия решения о приоритете новой функции по отношению к другим функциям, определенным для реализации в данной версии. Если новая функция является критической, она должна быть включена в реализацию. Заказчик совместно с командой проекта должен определить, какие функции следует исключить из реализации или, по крайней мере, как понизить их приоритеты с соответствующим уменьшением ожиданий. Но если функция является важной, а не критической, команда может исходить из того, что в про-

цессе разработки будет ясно, можно ли данную функцию реализовать в версии.

Управление проектами комплексов и компонентов для сложных программных продуктов – это особый вид инженерной деятельности, включающий: постановку задач, подготовку эталонов, планирование, организацию и стимулирование отдельных специалистов, контроль хода работ и использования ограниченных ресурсов при создании сложных систем и их составляющих. Критическим параметром управления такими проектами обычно является ***ограниченное время***. Основная проблема целевого управления такими работами – сводить воедино усилия коллектива исполнителей – специалистов разной квалификации, подрядчиков и субподрядчиков, добиваясь, чтобы они выступали при создании компонентов, комплекса программ и всей системы как ***команда***, а не как разрозненная группа независимых, функциональных специалистов. В результате должны обеспечиваться ***концептуальная целостность системы*** и высокое качество решения ее главных целей и функций ***при сбалансированном использовании ресурсов*** на реализацию всей совокупности задач и компонентов.

При разработке требований на сложный комплекс программ, кроме базовых функциональных, ресурсных и временных ограничений, могут быть ***общесистемные ограничения,*** которые также приходится учитывать:

- экономические – финансовые или бюджетные ограничения; себестоимость и ценообразование; проблемы лицензирования;
- политические – внешние или внутренние политические ограничения предприятия, влияющие на потенциальное решение; проблемы в отношениях между подразделениями и с другими предприятиями;
- технические ограничения проекта в выборе технологий; требования работать в рамках существующих платформ или технологий; запрет использования новых технологий; ограничения использовать закупаемые пакеты программной инженерии;
- системные требования обеспечивать совместимость с существующими решениями и системами; применение определенных операционных систем и внешней среды комплекса программ;

- эксплуатационные ограничения информационной среды; правовые или юридические ограничения; специфические требования безопасности; ограничения требованиями стандартов.

Требования как эталон к программному комплексу должны детализировать и конкретизировать описания функций до уровня, позволяющего разработчикам вести производство тестируемых модулей, компонентов и всего комплекса, которые могут быть проверены на корректность их реализации. Функции и основные характеристики сложных комплексов программ условно можно отразить многомерными ***пространствами свойств и значений***, контроля и обеспечения их реализации. Особенности, соответствие и покрытие этих многомерных пространств, их взаимодействия при различных задачах применения упрощенно можно представить как три пространства, ***функции и характеристики*** которых, используются и взаимодействуют в следующих целях и назначении:

- исходные, ***утвержденные требования – эталоны*** к функциям и характеристикам программного комплекса, согласованные с разработчиками в виде конкретных документов, в соответствии с которыми разработчики обязаны создать и обеспечить применение программного продукта пользователями или в составе системы;
- ***реально реализованные*** разработчиками функции и характеристики программного комплекса, которые обычно не могут полностью и абсолютно точно соответствовать исходным эталонным требованиям, достоверно ***не известны*** заказчику, разработчикам и пользователям и не отражены документами;
- реальные функции и характеристики программного комплекса, которые ***практически используются*** пользователями и/или системой в соответствии с эксплуатационной документацией и могут не совпадать с исходными реализованными требованиями вследствие превышения их значений или не полного их использования.

Функции и характеристики в перечисленных областях обычно не совпадают, но в некоторой степени пересекаются и в совокупности отражают возможные ситуации, которые полезно иметь в виду при организации и планировании тестирования. Функции и характеристики программного продукта, которые достигаются реально разработчиками, обычно достоверно не известны или не используются заказчиком и пользователями, или могут находиться за пределами заданных исходных требований. Наибольшее значение для реального обес-

печения качества программного продукта имеет ***полнота соответствия*** исходных требований к функциям и характеристикам, пространствам выполненных проверок при тестировании, адекватными им по содержанию и сценариям тестами. Программный менеджер и/или системный инженер должен быть знаком с основными способами проектирования сложных систем, знать, ***как перевести расплывчатые требования и пожелания заказчика системы в четкое техническое задание***, и уметь разговаривать с заказчиком и потенциальными пользователями системы на языке предметной области, а не на профессиональном программистском жаргоне. Такие способности требуют, в свою очередь, гибкости и открытости, чтобы выделять сущность предметной области программных комплексов.

В ***требованиях к продукту и процессу*** должно проводиться разграничение как свойств продукта, который необходимо получить, и процесса, с помощью которого продукт будет создаваться. При этом ряд требований может быть изложен неявно и программные требования могут порождать требования к процессу. Они устанавливают основные соглашения между пользователями (заказчиками) и разработчиками в отношении того, ***что должна делать система*** и чего от нее не стоит ожидать. Документ должен включать процедуры проверки получаемого программного продукта на соответствие предъявляемым ему требованиям, характеристики, определяющие качество и методы его оценки, вопросы безопасности и другие свойства. В то же время, существуют полуформальные и формальные методы и подходы, используемые для спецификации программных требований. В любом случае, задача состоит в том, чтобы программные требования были ясны, связи между ними прозрачны, а содержание спецификаций не допускало разночтений и интерпретаций, способных привести к созданию программного продукта, не отвечающего потребностям заинтересованных лиц.

При формировании требований к программному комплексу и компонентам менеджеры разработки и тестирования должны осуществлять согласованные действия в соответствии с принятой ***политикой и процедурами создания проекта***:

- определять функциональные границы компонента или комплекса в терминах его поведения и предусмотренных свойств внешней среды;

- определять каждую функцию, которая должна быть реализована в программном комплексе и компоненте для обеспечения их корректного применения;
- определять необходимые ограничения по реализации и тестированию, обусловленные требованиями или неизбежными ограничениями системы;
- определять технические и потребительские характеристики комплекса и компонентов, позволяющие оценивать технические результаты функционирования;
- задавать нефункциональные системные требования, в соответствии с которыми определяются риски и параметры программного комплекса, связанные с критическими показателями: надежностью, безопасностью, производительностью;
- на протяжении всего жизненного цикла вести учет состояния совокупности требований вместе с их обоснованиями, изменениями, связанными решениями и допущениями.

Каждое положение ***требований к программному комплексу и компоненту должно проверяться и верифицироваться*** для установления его корректности, полноты, непротиворечивости, совместимости с требованиями других компонентов, реализуемости, тестируемости и проверяемости при испытаниях. Требуется удостоверять, что требования являются, с одной стороны, ***необходимыми и достаточными*** для удовлетворения при применении компонента, а с другой – необходимыми и достаточными входными данными для других процессов и компонентов, в частности, ***для проектирования функций и архитектуры комплекса программ***.

Формирование требований компонентов и модулей путем декомпозиции функций комплексов программ

Многочисленное дублирование, по существу, одних и тех же программных компонентов на подобных или разных платформах сопряжено со значительными нерациональными затратами на их разработку и с увеличением длительностей создания систем. Использование методического, технологического, алгоритмического и программного задела из предшествующих проектов обеспечивает многократное повышение производительности труда разработчиков сложных комплексов программ, сокращение сроков их создания и высокое качество. На ***экономические характеристики*** проектирования и про-

изводства программных комплексов весьма сильно оказывает влияние возможность использования ***апробированного задела*** из предыдущих реализованных проектов. Результаты системного анализа, применение функциональных и информационных моделей предметной области, формализация спецификаций требований, функциональная ***декомпозиция программных комплексов*** и последовательная детализация проектов позволяют применять готовые технические решения в различных формах и сочетаниях. В ряде случев имеется возможность ***повторного использования компонентов*** (ПИК) или модулей программ на разных уровнях описания, относящихся к разным этапам их жизненного цикла. Возникла проблема разработки функционально законченных программных компонентов и модулей, потенциально готовых, к многократному применению в различной внешней и операционной среде, а также в различных сочетаниях их взаимодействия в комплексах программ (см. рис. 1.4).

Программные компоненты и модули для производства сложных комплексов программ могут создаваться ***двумя методами***:

- в процессе системного проектирования конкретного комплекса программ и его ***последовательной декомпозиции*** на функциональные задачи и далее на небольшие ***уникальные*** программные компоненты и модули, которые могут иметь произвольную архитектуру и интерфейсы для определенного проекта;
- ***путем поиска, подбора и повторного использования*** готовых апробированных компонентов и модулей, созданных для предшествовавших проектов, с учетом возможности их эффективного использования в других комплексах программ за счет ***унифицированной архитектуры и интерфейсов***.

Декомпозиция функций комплекса программ и выявление особенностей при создании, программировании и тестировании его конкретных компонентов основывается на разбиении общей цели и функций проекта на несколько промежуточных целей и этапов, каждый из которых также можно разделить. Этот процесс можно повторять до тех пор, пока каждая цель и функция не станет достаточно мелкой для ее полного ***представления и разработки в виде программного модуля или компонента***. Разбиение общей цели проекта – ***декомпозиция производственных работ*** может начинаться с построения символического дерева, корень которого помечен основной целью и функцией проекта. Каждый узел такого дерева можно раз-

бить на более мелкие производственные компоненты. Такую декомпозицию можно повторять до тех пор, пока каждый компонент не будет представлять часть работы, которую руководитель сможет оценить по размеру, сложности выполнения и необходимым ***ресурсам.***

Целью декомпозиции комплекса программ является идентификация всех компонентов, видов и объектов производства, которые могут выполнять ***отдельные специалисты*** по ходу реализации проекта. Для сложных комплексов программ экономические характеристики обычно трудно прогнозировать для отдельных компонентов и операций вследствие разнообразия сложности и размеров компонентов, индивидуальных характеристик создающих их специалистов и множества других производственных факторов. Декомпозиция работ, компонентов и специалистов обеспечивает руководителей базой для решения задач планирования и управления обозримыми и контролируемыми частями. После того, как последние определены, их можно использовать в качестве элементов производственных работ для экономической оценки, распределения между специалистами с учетом их квалификации, с соответствующей запланированной продолжительностью выполнения, именем ответственного лица, датой начала и окончания работы. Декомпозицию работ и компонентов можно использовать как исходную информацию в процессе календарного планирования всего комплекса программ. При составлении календарного плана может производиться упорядочивание компонентов для обеспечения своевременного и скоординированного решения комплекса производственных задач.

В современных автоматизированных технологиях создания и совершенствования сложных комплексов программ с позиции обеспечения их качества можно выделить ***базовые методы и средства***, позволяющие:

- создавать программные модули и функциональные компоненты высокого, гарантированного качества;
- предотвращать дефекты проектирования за счет систем обеспечения качества, эффективных технологий и инструментальных средств автоматизации всего жизненного цикла компонентов и комплексов программ;
- предотвращать, обнаруживать и устранять различные дефекты и ошибки проектирования, разработки и сопровождения компо-

нентов путем верификации и систематического тестирования на всех этапах жизненного цикла комплекса программ;

- удостоверять достигнутые значения качества функционирования программного комплекса и компонентов в процессе их испытаний и сертификации перед передачей заказчику и в регулярную эксплуатацию пользователям.

Процессы производства сложных комплексов программ основаны на исходных принципах ***модульности***. Процессы являются модульными в том смысле, что их исходные компоненты: ***невелики, обозримы, строго связаны и взаимоувязаны***. Число интерфейсов между процессами и компонентами желательно сводить к минимуму. В принципе каждый процесс предназначен для реализации уникальной функции компонента (модуля) в жизненном цикле комплекса программ, может привлекать и использовать другие процессы и компоненты для выполнения более сложной специализированной функции.

Унификация и структурирование процессов декомпозиции комплексов программ оправдана, если она обеспечивает экономию времени производства и/или применения продукта. ***Архитектура процессов и компонентов комплекса программ*** – это структура, организующая процессы взаимодействия компонентов, используемых при построении сложных функций, образующих версии программных комплексов. Каждая часть архитектуры обычно включает множество компонентов, даже в относительно простых комплексах. Разработка унифицированной структуры компонентов особенно целесообразна для версий программных комплексов, когда затраты могут эффективно окупаться при производстве множества последовательных вариантов – версий продуктов. Чтобы быть полезным, формальные процессы и компоненты комплекса необходимо адаптировать к специфике функций конкретного проекта.

Основные требования структурирования архитектуры комплексов программ можно объединить в группы, которые устанавливают:

- стандартизированную структуру (архитектуру) целостного построения комплекса программ определенного класса;
- унифицированные требования структурного построения функциональных программных компонентов и модулей;

- стандартизированную структуру баз данных, обрабатываемых программами;
- унифицированные требования структурного построения информационных модулей, заполняющих базу данных;
- унифицированные требования к организации и структурному построению межмодульного интерфейса программных компонентов;
- унифицированные требования внешнего интерфейса и взаимодействия комплекса программ и базы данных с внешней средой, с операционной системой и другими типовыми средствами организации вычислительного процесса, защиты и контроля.

Таким образом, для обеспечения эффективного производства и сокращения затрат необходимо формулировать и соблюдать ряд принципов и правил структурного построения и повторного применения программных компонентов и модулей (см. рис. 1.4). Эти принципы и правила могут иметь особенности в различных проблемно-ориентированных областях. Однако их формализация и выполнение в конкретных проектах обеспечивают значительное ***снижение трудоемкости и длительности производства*** программных продуктов и их версий. Потеря гибкости архитектуры комплексов программ, некоторое возрастание ресурсов, необходимых для их реализации, обычно полностью компенсируются ***улучшением экономических характеристик*** процессов проектирования и производства продуктов.

Структурное проектирование программных комплексов должно быть ***основано на модульном принципе***. Многоуровневое, иерархическое построение сложных программных комплексов позволяет ограничивать и локализовать на каждом из уровней сложности соответствующие ему компоненты. ***Нижнему иерархическому уро***вню представления программ соответствуют программные и информационные модули (модули данных). Эти компоненты (например, 10 – 100 модулей) объединяются в группы на ***среднем уровне компонентов программ*** определенного функционального назначения с автономной целевой задачей. Несколько групп функциональных программ образует целостный функциональный ***комплекс программ высокого уровня*** определенной системы. В сложных случаях возможно создание программного продукта из нескольких взаимодействующих функциональных комплексов программ высокого уровня. Всем иерархическим системам (в частности, программным комплексам) присущ ***ряд свойств и требований***, важнейшими из которых являются:

- вертикальная соподчиненность, заключающаяся в последовательном упорядоченном расположении взаимодействующих компонентов, составляющих программный комплекс;
- право вмешательства и приоритетного воздействия компонентов верхнего уровня сверху вниз на компоненты нижних уровней;
- взаимозависимость действий компонентов верхних уровней от реакций на воздействия и от функционирования компонентов нижних уровней, информация о которых передается верхним уровням.

В результате в иерархических структурах комплексов программ образуются ***два потока взаимодействий*** между компонентами разных уровней: ***сверху вниз*** – координирующие и управляющие воздействия верхних уровней на компоненты нижних уровней и ***снизу вверх*** – информация о состоянии и реализации предписанных сверху функций компонентами нижних уровней. Координируемые компоненты обычно имеют некоторую автономность поведения и подготовки локальных функциональных решений. Степень автономности компонентов и интенсивность координирующих воздействий устанавливаются в результате ***компромисса*** при выделении числа и размера компонентов ***иерархических уровней***. Взаимодействие компонентов в пределах уровня целесообразно максимально ограничивать, что позволяет упростить общее координирование компонентов и проводить его только по вертикали.

Характеристики внешней среды применения программного продукта в значительной степени определяют архитектуру и структуру применяемой операционной системы, средств контроля и организации вычислительного процесса. При разработке серии версий для некоторой прикладной области целесообразно унифицировать внешний интерфейс и операционную систему. Это обеспечивает многократное использование одних и тех же организующих программ, дисциплинирует архитектурное построение всего комплекса и способствует унификации межмодульного интерфейса. Переход на новую операционную систему, так же как и переход на реализующую ЭВМ другого типа, может приводить к необходимости изменения структурного построения комплекса программ и базы данных.

Изменения, порожденные адаптацией такого рода, часто довольно существенны и дают результат, ***мало похожий на первичную структуру***. Структурирование должно использовать детальную информацию о проекте: правила конфигурирования компонентов

комплекса, наборы инструкций разработчикам, программистам и тестировщикам, рекомендации на формирование документации и программные интерфейсы компонентов. Кроме того, необходимо выделять ключевых специалистов для выполнения важнейших шагов процесса интеграции компонентов, отслеживания дефектов или сборок, фиксации изменений, контроля стиля кодирования и взаимного просмотра кода, а также многие другие детали, специфичные для взаимодействия компонентов любого сложного проекта.

При организации структуры сложного комплекса программ, создаваемого большим коллективом специалистов, естественно возникает проблема оценки и упорядочивания ***целесообразных размеров*** модулей и компонентов на разных иерархических уровнях. Рациональные размеры программных модулей (ПМ) могут быть ограничены удобством и обозримостью разработки текстов программ и их тестирования отдельными специалистами. Хотя у некоторых ***программистов «виртуозов»*** есть тенденция писать монолитные модули размером во многие сотни строк, однако при этом возникают трудности тестирования и обеспечения высокой корректности таких программ. Экспериментально установлено, что во многих программах управления и обработки информации – линейные участки программ между предикатами – узлами с ветвлением ***в среднем*** составляют около 5 – 10 строк. Поэтому число маршрутов исполнения программ и соответствующее число тестов, необходимых для их проверки, возрастает не пропорционально числу строк в программе, а значительно быстрее (почти квадратично, см. лекцию 2.1). Уже при ста строках в программе (10 – 20 предикатов – узлов ветвления) для ее тестирования может потребоваться более 100 тестов. Таким образом, при разработке ПМ целесообразно учитывать рациональное ***ограничение размеров модулей на уровне трехсот строк текста***, что соответствует приблизительно тридцати альтернативам в таких программах. При этом для полного покрытия таких ПМ тестами необходимо задавать до 1000 условий в тестах, что обычно достаточно трудно или невозможно реализовать практически в ограниченное время. В среднем полное тестирование программ с 30-ю вершинами ветвления производится тестами с суммарной сложностью около 300 – 500 узлов – предикатов.

Программные компоненты высокой сложности целесообразно делить на более простые и легче тестируемые модули, что во многих

случаях делается программистами интуитивно. Анализ числа тестов для большой реальной выборки программных модулей в комплексе программ, создаваемом коллективом специалистов, показал, что основная часть модулей содержала около 200 строк (до 20 – 30 узлов ветвления). Это обычно устанавливалось средними специалистами вследствие психологической сложности разработки и тестирования более крупных модулей. Поэтому в ряде предприятий при разработке ПМ был ***рекомендован рациональный размер программ модулей*** в пределах 100 – 300 строк текста, для полного тестирования которых достаточно использовать 10 – 50 тестов с суммарным числом условий ветвления до 100. При превышении рекомендуемых размеров ПМ их трудно протестировать полностью и ***целесообразно делить программистам на более мелкие компоненты – модули***, доступные для практически полного покрытия тестами, размеры которых регламентировать инструкциями проекта.

Каждый модуль должен рассматриваться с точки зрения ***ответственности определенных личностей специалистов – программиста и тестировщика***. Коллектив или личность, выполняющая разработку, несет персональную ответственность за весь данный процесс и компонент, даже если выполнение отдельных задач поручено другому предприятию. Принцип фиксирования ответственности лиц в архитектуре, компонентах и процессах производства программного комплекса облегчает применение стандартов и ***оценку характеристик*** части конкретного проекта, в который может быть вовлечено множество специалистов.

В процессе эксплуатации программного продукта у каждого пользователя могут появляться некоторые претензии к функционированию, которые квалифицируются им как ошибки или дефекты базовой или собственной, адаптированной версии программного продукта. От пользователей или заказчиков могут поступать также предложения по внесению изменений в базовую версию для улучшения эксплуатационных характеристик и расширения функциональных возможностей. Аналогичные предложения могут поступать от разработчиков комплекса программ. Для решения таких задач разработаны и активно применяются стандартизированные методы, методики и средства автоматизации регламентированного ***сопровождения и управления конфигурацией компонентов и комплексов программ***. Они позволяют представить отдельным специалистам и руководите-

лям состояние проекта и его компонентов в любой момент времени и не допускать хаоса при коллективной модификации программ и данных. Дисциплина сопровождения и конфигурационного управления компонентами в значительной степени ***определяет экономические характеристики*** сложного программного комплекса, его качество, длительность создания, применения и конкурентоспособность (см. лекцию 2.7).

Возрастание сложности и ответственности современных задач, решаемых крупными системами, а также возможного ущерба от недостаточного качества программного продукта, значительно повысило актуальность освоения методов ***стандартизированного описания требований и оценивания характеристик качества*** производственных процессов, компонентов и продуктов на различных этапах жизненного цикла. Выявилась необходимость систематизации реальных характеристик качества, применения стандартов для выбора из них необходимой номенклатуры и требуемых значений для конкретных производств комплексов программ.

Лекция 1.3

ТРЕБОВАНИЯ К ФУНКЦИЯМ И ХАРАКТЕРИСТИКАМ КАЧЕСТВА КОМПЛЕКСОВ ПРОГРАММ

Особенности требований заинтересованных лиц к функциям и характеристикам комплексов программ

При разработке требований к комплексу программ необходимо ***выделять и ранжировать по приоритетам заинтересованных лиц***, которым необходимы определенные функции и показатели качества программного комплекса с учетом их специализации и профессиональных интересов – рис. 1.5. Широкая номенклатура характеристик, представленная в стандарте **ISO 9126,** определяет разнообразные требования, из которых следует селектировать и выбирать те, которые необходимы ***с позиции потребителей этих данных***:

- ***заказчиков***, для которых важно регламентировать и оценивать программный продукт по значениям требований к функциям и характеристикам, заданным и утвержденным в контракте, техническом задании и спецификациях требований и определяющих, прежде всего, назначение, функции и сферу применения программного комплекса;
- ***пользователей***, для которых необходима функциональная пригодность, корректность, надежность и другие показатели качества при оперативном использовании комплекса программ по основному назначению;
- ***разработчиков***, для которых важны: ясность и конкретность описаний требований к функциям и характеристикам программного комплекса, его возможная архитектура и интерфейсы между компонентами и с внешней средой;
- ***специалистов сопровождающих и модифицирующих*** комплекс программ, которые отдают приоритет характеристикам, поддерживающим сопровождение и конфигурационное управление версиями комплекса и его компонентов;

Требования к функциям и характеристикам качества комплексов программ должны включать:

- требований заинтересованных лиц к функциям и характеристикам комплексов программ;
- функциональные требования к сложному комплексу программ:
  - системные основы требований к сложному комплексу программ;
  - требования к назначению и характеристикам функциональной пригодности комплекса программ;
  - стратегию управления функциональными требованиями к комплексу программ;
  - планирование разработки и реализации требований к функциональным характеристикам программного комплекса;
  - требования к архитектуре и качеству программного комплекса в составе системы;
  - ограничения ресурсов проектирования, производства и применения программного комплекса;
- общие требования к качеству функционирования программных комплексов:
  - функциональная пригодность и конструктивные характеристики комплексов программ;
  - внутреннее качество проектирования и производства и внешнее качество функционирования и применения;
  - требования к качеству функциональных характеристик и требования к структурным характеристикам комплексов программ реального времени;
- ограничения ресурсов для реализации требований к программному комплексу реального времени;
- требования к корректности, надежности и безопасности программных комплексов;
- требования к эффективности использования ресурсов ЭВМ программным комплексом в реальном времени;
- проверку корректности функциональных требований к комплексам программ.

Рис. 1.5.

- ***лицам, ответственным за инсталляцию***, внедрение и реализацию программного комплекса в различных аппаратных и опера-

ционных средах, для которых наиболее важны характеристики обеспечения мобильности.

Приоритеты потребителей при формировании требований к комплексу программ отражаются не только на выделении важнейших для них критериев и ранжировании приоритетов других характеристик, но также на возможности исключении из анализа некоторых характеристик качества, которые для данного потребителя не имеют значения. Ранжирование может детализироваться и изменяться в зависимости от функций комплексов и реальных ресурсов, доступных для обеспечения их жизненного цикла. Эти приоритеты должны обобщаться и учитываться заказчиком проекта при подготовке контракта и технического задания. После определения назначения и функций комплекса программ подготовка исходных данных и концепции проекта должны завершаться ***выделением номенклатуры приоритетных функций и характеристик качества***, имеющих влияние на функциональную пригодность для определенных потребителей.

Требования к значениям функций и характеристик программного продукта должны быть предварительно проверены разработчиками на их реализуемость с учетом доступных ресурсов конкретного проекта и при необходимости откорректированы по составу и значениям с учетом рисков. При ограниченности ресурсов проекта распределение приоритетов должно становиться более строгим и могут снижаться приоритеты характеристик, для реализации которых ресурсов недостаточно.

Команда разработчиков должна применить методы и процессы для того, чтобы ***понять решаемую проблему заказчика до начала разработки программного комплекса***. Для этого следует использовать ***анализ, выявление и освоение профессиональной проблемы и интересов заказчика***:

- достигнуть соглашения между заказчиком и разработчиком по определению проблемы, целей и задач проекта;
- выделить основные причины – проблемы, являющиеся их источниками и стоящие за основной проблемой проекта системы и комплекса программ;
- выявить заинтересованных лиц и пользователей, чье коллективное мнение и оценка в конечном итоге определяет успех или неудачу проекта;

- определить, где приблизительно находятся область и границы возможных решений проблем;
- понять ограничения, которые будут наложены на проект, команду и решения проблем.

Понимание потребностей пользователей необходимый организационный этап, так как разработчики редко получают совершенные спецификации требований к создаваемой системе, они должны сами ***добывать*** информацию, необходимую им для успешной работы. Термин ***выявление требований*** точно отражает данный процесс, в котором разработчики должны играть активную роль. Чтобы помочь команде решить эти проблемы, лучше понять потребности пользователей и других заинтересованных лиц, целесообразно использовать ***методы***:

- интервьюирования и анкетирования – создание структурированного интервью возможных требований;
- проведение интервью с несколькими пользователями и/или заинтересованными лицами;
- подведение итогов совокупности интервью, формулирование набора наиболее часто упоминавшихся функциональных и архитектурных потребностей заказчика и пользователей;
- совещания, посвященные анализу и синтезу требований – формулирование и определение целей программного продукта; ознакомление с ними всех участников проекта и установление, что они с ними согласны; если это не так, следует остановиться и уточнениями добиться согласия; обязательно убедиться в согласии заказчиков с выделенными требованиями;
- анализ иллюстративных прецедентов в приложении к концепции формируемых требований, чтобы их функции были наглядны и понятны;
- по возможности выявление или создание временных прототипов на основе первичных требований.

Хотя ни один из методов не является универсальным, каждый из них позволяет лучше понять потребности пользователей и тем самым превратить неясные требования в требования, которые сформулированы и понятны всем заинтересованным лицам. Каждый из этих методов эффективен в определенных ситуациях, однако целесообразно отдавать предпочтение групповым совещаниям, посвященным требованиям.

Формирование функциональных требований к сложным комплексам программ

При планировании, разработке и реализации ***требований к характеристикам программного средства*** необходимо, в первую очередь, учитывать следующие факторы:

- функциональную пригодность (функциональность) конкретного проекта программного комплекса;
- возможные конструктивные характеристики качества комплекса программ, необходимые для обеспечения функциональной пригодности;
- доступные ресурсы для создания и обеспечения всего жизненного цикла программного средства с требуемым качеством.

Эти факторы целесообразно применять последовательно, итерационно на каждом из основных этапов жизненного цикла комплекса программ. При первоначальном определении требований к функциональной пригодности и к конструктивным характеристикам качества, заданные заказчиком ограничения ресурсов не всегда могут учитывать ряд особенностей проекта, что обусловит недопустимое снижение (или завышение) требований к некоторым характеристикам качества. Кроме того, возможно, что некоторые характеристики противоречивы или принципиально нереализуемы в данном проекте. В результате ***не сбалансированные требования*** к качеству и доступные ресурсы проявятся как риски – ущерб в виде потерь в качестве или в перерасходовании ресурсов. Для устранения или снижения рисков до допустимых пределов потребуется изменение требований к функциональной пригодности и/или к конструктивным характеристикам. Таким образом, целесообразно анализ и разработку требований к качеству комплекса программ проводить в два этапа: предварительно ***максимизируя функциональную пригодность*** и конструктивные характеристики качества, а затем ***минимизируя риски*** снижения требуемого качества или используемых ресурсов.

Требования к назначению и характеристикам функциональной пригодности и к конструктивным характеристикам качества существенно зависят от назначения и функций комплексов программ. Так, например, при проектировании:

- систем управления объектами в реальном времени большое значение имеет время реакции (отклика на задание), защищенность,

корректность и надежность функционирования стабильного комплекса программ и менее важно может быть качество обеспечения сопровождения и конфигурационного управления, способность к взаимодействию и практичность;

- для административных систем, кроме корректности, важно обеспечивать практичность применения, комфортное взаимодействие с пользователями и внешней средой, и может не иметь особого значение эффективность использования вычислительных ресурсов и обеспечение мобильности комплекса программ;
- для пакетов прикладных программ вычислений или моделирования процессов можно не всегда учитывать их мобильность, защищенность и временную эффективность, но особенно важна может быть корректность и точность расчетов.

Системная эффективность – функциональная пригодность комплекса программ может быть описана количественно или качественно в виде набора полезных свойств и характеристик программного средства, их отличий от имеющихся у других комплексов программ, а также источников возможной эффективности. Она определяет назначение, основные функции и требования заказчика, какие задачи должны решаться для удовлетворения пользователей, а дополнительные, конструктивные характеристики качества – как и при каких условиях заданные функции могут выполняться с требуемым качеством. В результате может быть формализована цель использования и набор главных характеристик, требований заказчика и пользователей при заказе или приобретении программного продукта, а также предполагаемая его сфера назначения и применения. Полнота и точность представления этих характеристик является исходной для прослеживания реализации всех последующих, производных свойств и качества функциональной пригодности комплексов программ (см. рис. 1.4).

Данное требование связано с тем, ***какие*** функции и задачи должен решать программный продукт для удовлетворения потребностей пользователей, в то время как другие, конструктивные требования главным образом связаны с тем, ***как и при каких условиях*** заданные функции могут выполняться с требуемым качеством**.** ***Функциональная пригодность*** – это набор и описания атрибутов, определяющих ***назначение, основные, необходимые и достаточные функции программного комплекса***, заданные техническим заданием и спецификациями требований заказчика или потенциального пользователя.

Номенклатура и значения всех остальных показателей качества непосредственно определяются требуемыми функциями программного комплекса и в той или иной степени влияют на выполнение этих функций. Поэтому выбор функциональной пригодности, подробное и корректное описание ее свойств являются основными исходными данными для установления требуемых значений всех остальных стандартизированных показателей качества. Атрибутами этой характеристики могут быть функциональная полнота решения заданного комплекса задач, степень покрытия функциональных требований спецификациями и их стабильность при совершенствовании, число и полнота реализуемых требований заказчика. Такими атрибутами могут быть: функциональная адекватность программ документам и декларированным требованиям, утвержденным заказчиком; степень покрытия тестами исходных требований; полнота и законченность реализации этих требований; точность выполнения требований детальных спецификаций на функциональные компоненты.

Для сложных программных комплексов необходима ***стратегия управления требованиями***. Для этого применяется информационная ***иерархия***; она начинается с потребностей пользователей, описанных с помощью функций системы, которые затем превращаются в более подробные требования к программному комплексу, выраженные посредством прецедентов или традиционных форм описания и стандартизированных характеристик. Эта иерархия отражает уровень абстракции при рассмотрении ***взаимосвязи области требований и области решений***. В требованиях к комплексу программ следует указывать, ***какие функции*** должны осуществляться, а ***не то, как*** они могут реализоваться. Нужно использовать набор характеристик для установления и оценки качества комплекса и содержащихся в нем функциональных компонентов. Если необходимо, документация требований может дополняться одним или несколькими более формальными либо более структурированными методами спецификации требуемого качества комплекса программ.

Функциональные требования – определяют поведение программного комплекса, который должен быть создан разработчиками для предоставления возможности выполнения заказчиком и/или пользователями своих обязанностей в контексте пользовательских требований. Выбор требований к характеристикам при проектировании программных комплексов начинается с ***определения исходных***

данных. Для корректного выбора и установления требований к характеристикам качества, прежде всего, необходимо определить основные особенности программной системы. На основе этих данных должен формироваться общий ***набор требуемых характеристик, свойств, их мер и значений качества для определенных потребителей в жизненном цикле программного комплекса***:

- цели создания программного продукта и назначение комплекса функциональных задач;
- общие требования к функциям и характеристикам задач программного комплекса;
- перечень объектов и характеристик внешней среды применения программного комплекса (технологических объектов управления, подразделений предприятия и т.п.), при управлении которыми должен решаться комплекс задач;
- периодичность и продолжительность решения комплекса задач;
- связи и взаимодействие комплекса задач с внешней средой и другими компонентами системы;
- распределение функций между персоналом, программными и техническими средствами при различных ситуациях решения требуемого комплекса функциональных задач.

Проекты, как правило, инициируются заказчиком с ***объемом функциональных возможностей, превышающим*** тот, который разработчик может реализовать, обеспечив приемлемое качество при заданных ресурсах. Тем не менее, необходимо ограничиваться, чтобы иметь возможность предоставить в срок ***достаточно целостный и качественный программный продукт***. Существуют различные методы задания очередности выполнения (приоритеты) требований и понятие базового уровня – совместно согласованного представления о том, в чем будут состоять ключевые функции системы, требований, задающих ориентир для принятия решений и их оценки.

Если ***масштаб проекта и сопутствующие требования заказчика превышают реальные ресурсы***, в любом случае придется ограничиваться в функциях и качестве комплекса программ. Поэтому следует определять, что ***обязательно должно*** быть сделано в первой или очередной версии системы и программного продукта при имеющихся ресурсах проекта. Для этого приходится вести переговоры. Нельзя ожидать, что данный процесс полностью решит проблемы

масштаба и требований к программному продукту. Но такие шаги окажут заметное воздействие на размеры проблемы, позволят разработчикам сконцентрировать свои усилия на критически важных подмножествах требований и функций и итерационно, в несколько этапов, предоставить последовательные версии высококачественных систем, удовлетворяющих или даже превосходящих основные ожидания пользователей. Привлечение заказчика к итерационному решению проблемы управления масштабом и функциями повышает взаимные обязательства сторон, способствует росту взаимопонимания и доверия между заказчиком и разработчиками. Имея достаточное определение функций продукта (концепцию) и сократив масштаб проекта до разумного уровня, можно надеяться на успех в следующих версиях программного комплекса.

Общие требования к качеству функционирования сложных программных комплексов

Базовые международные стандарты определяют основные требования к качеству в жизненном цикле комплексов программ (см. Приложение 1). Непосредственно к характеристикам качества программных средств относится ***группа стандартов***, в которую входят нормативные документы, регламентирующие формализацию ***нефункциональных требований*** к метрикам качества комплексов программ, методы и процессы их выбора и измерения. Эта группа является важнейшей для достижения и гарантирования высокого качества программного продукта на всех этапах жизненного цикла. Тщательное ***специфицирование требований к качеству программного продукта – ключевой фактор обеспечения их адекватного применения***. Это может быть достигнуто на основе выделения и определения подходящих характеристик качества с учетом целей использования и функциональных задач комплекса программ (см. рис. 1.5).

Основой для формирования требований к характеристикам комплексов программ является анализ свойств, отражающих качество их функционирования. При этом под ***качеством функционирования понимается*** множество свойств, обусловливающих пригодность программного продукта обеспечивать надежное и своевременное представление требуемой информации заказчику и/или потребителю для его дальнейшего использования по назначению. В соответствии с принципиальными особенностями конкретного комплекса программ

при проектировании должны выбираться номенклатура и значения показателей качества, необходимых для его эффективного применения пользователями, которые отражаются в технической документации и в ***спецификациях требований*** на конечный программный продукт.

Формализация требований к качеству программных продуктов могут проводиться с двух позиций: с ***позиции положительной*** обеспечения эффективности и непосредственной адекватности их характеристик качества – назначению, целям создания и применения программного продукта, а также с ***негативной позиции*** возможного при этом ущерба – допустимого риска от дефектов и недостатков качества при применении комплекса программ или системы. ***Характеристики качества и риски*** объектов и процессов обычно тесно связаны, на них влияют подобные внутренние и внешние факторы, которые с разных сторон отражаются на свойствах систем или комплексов программ. Требуемые и реальные характеристики качества влияют на риски, а остающиеся риски влияют на характеристики качества программного продукта. Повышению требований к характеристикам качества проекта обычно ***сопутствует*** снижение его рисков и наоборот, сокращение рисков способствует улучшению характеристик качества и повышению функциональной пригодности. Их сбалансированное взаимовлияние должно обеспечивать ***требуемую функциональную пригодность*** программного продукта. Методы и системы управления качеством близки к методам анализа и управления рисками проектов комплексов программ, они должны их дополнять и совместно способствовать совершенствованию функциональной пригодности программных продуктов и систем на их основе.

Системная эффективность целевого применения программных продуктов в стандарте **ISO 9126** определяется степенью удовлетворения потребностей определенных лиц – заказчиков и/или пользователей, которые, в ряде случаев, желательно измерять экономическими категориями: прибылью, стоимостью, трудоемкостью, предотвращенным ущербом, длительностью применения и т.п. В стандартах эта эффективность отражается основной, обобщенной характеристикой качества – ***функциональная пригодность***. В связи с тем, что ее абсолютную величину обычно трудно измерить непосредственно и количественно, то по ряду показателей необходима и возможна каче-

ственная оценка свойств и достоинств при применении программного продукта.

Улучшение каждой ***нефункциональной – конструктивной характеристики качества*** требует некоторых затрат ресурсов, которые в той или иной степени должны отражаться на основной характеристике качества – на функциональной пригодности. Требования к конструктивным характеристикам имеют значение для проекта постольку, поскольку они обеспечивают требуемое качество реализации основного назначения и функций комплекса программ. При выборе требований конкретных мер и шкал конструктивных характеристик качества следует учитывать возможные затраты ресурсов на их достижение и на результирующее повышение функциональной пригодности, желательно, в сопоставимых экономических единицах, в тех же мерах и масштабах. Поэтому для каждого проекта необходимо ранжировать характеристики и их атрибуты (приоритеты) и выделять, прежде всего, те требования, которые могут в наибольшей степени улучшить функциональную пригодность для конкретных целей.

Стандарты описывают метрики качества комплексов программ на основе их внутренних атрибутов и внешнего поведения системы. Эти типы метрик применимы при определении требований к качеству и целей проекта, включая промежуточные продукты. Общее представление о качестве стандартами рекомендуется отражать тремя взаимодействующими и взаимозависимыми ***группами показателей***, характеризующими:

- внутреннее качество комплекса программ, проявляющееся в процессе проектирования и производства;
- внешнее качество, заданное требованиями заказчика;
- качество при использовании в процессе нормальной эксплуатации комплекса программ и результативностью достижения потребностей пользователей с учетом затрат.

Ограниченные ресурсы для реализации требований функциональной пригодности могут негативно отражаться на конструктивных характеристиках: на надежности, безопасности, пропускной способности, качестве взаимодействия с внешней средой и с пользователями, качестве документации и других эксплуатационных факторах. Таким образом, ***требования к характеристикам качества проекта комплекса программ и допустимых рисков*** должны быть проанализированы и согласованы для установления в договоре между заказчи-

ками и разработчиками исходных компромиссных значений, пригодных для эффективной разработки комплекса программ с требуемыми функциями. Для решения этих задач необходимо управление характеристиками качества и управление рисками с целью обеспечении разработчиками требуемой заказчиком функциональной пригодности. При этом следует учитывать, что каждый вид ресурсов в реальных условиях ограничен и может варьироваться только в некотором диапазоне.

Наиболее общим видом ресурсов, используемых в жизненном цикле комплексов программ, являются ***допустимые финансово-экономические затраты***. При анализе требований качества этот показатель может применяться или как вид ресурсных ограничений, или как оптимизируемый критерий. При этом необходимо также учитывать затраты на разработку, закупку и эксплуатацию системы обеспечения качества, технологии и комплекса автоматизации разработки программ и баз данных, которые могут составлять существенную часть совокупной стоимости программного продукта.

Каждое требование к характеристикам качества и затратам ресурсов первоначально обычно ***анализируются независимо***, что может использоваться в качестве исходных данных для их сопоставления с отдельными характеристиками аналогичных комплексов программ, или для представления, как составляющей вектора в многомерном пространстве требований стандартизированных характеристик качества. Обычно заказчики и разработчики первоначально устанавливают требования к каждой характеристике без учета относительных затрат на их достижение, а также без детального анализа их совместного влияния на полную функциональную пригодность у потребителей. Это может приводить к значительным перекосам и ***несбалансированным значениям требований*** к отдельным, взаимосвязанным характеристикам качества, на которые не рационально используются ограниченные ресурсы проекта, или к не адекватно низким их значениям. В проектах сложных комплексов программ это может угрожать значительным повышением стоимости и/или снижением конкурентоспособности создаваемого программного продукта из-за недостаточного уровня отдельных показателей качества.

Характеристики качества комплексов программ имеют различные меры и шкалы, вследствие чего они в большинстве своем непосредственно ***не сопоставимы между собой***. Они предварительно выбираются и согласовываются с заказчиком при последовательном,

почти независимом анализе каждого атрибута качества в соответствии с их мерами и шкалами для последующего использования в контракте и техническом задании. Для обобщенного оценивания качества необходим учет относительного влияния каждого атрибута на функциональную пригодность. При этом не всегда учитываются ресурсы для их реализации в конкретном проекте комплекса программ. Это часто приводит к выдвижению ряда не рациональных требований, которые значительно отличаются либо по степени влияния на функциональную пригодность, либо по величине ресурсов, необходимых для их реализации. Для целенаправленного, эффективного управления требованиями сложного комплекса программ целесообразно иметь ***механизм объединения разнородных характеристик*** в некоторый интегральный показатель, отражающий их совокупное влияние на функциональную пригодность.

Требования к характеристикам комплексов программ, определяющие их функциональную пригодность, ***принципиально различаются на две группы***:

- ***требования к количественным, измеряемым, функциональным характеристикам***, непосредственно влияющим на оперативное функционирование и возможность применения программного продукта в системе, в которые входят требования к надежности, безопасности, производительности, допустимым рискам применения;
- ***требования к структурным характеристикам***, определяющим архитектуру комплекса программ, влияющие на возможности его модификации и сопровождения версий, на мобильность и переносимость на различные платформы, на документированность, удобство практического освоения и применения программного продукта вне оперативного функционирования.

Общие требования к системам и комплексам программ реального времени для автоматизации управления и обработки информации о динамических объектах состоят в следующем:

- в процессах управления решением задач и вычислениями в ЭВМ должно использоваться единое глобальное реальное время систем управления и обработки информации динамических объектов, а также внешней среды;
- потоки данных для обработки из внешней среды могут быть независимыми, несинхронными, различными по интенсивности, со-

держанию, реальному времени формирования и поступления для обработки;

- информация в сообщениях от источников и объектов внешней среды должна содержать реальное время, к которому относятся сообщения, их координаты и характеристики состояния;
- реальное время решения различных функциональных задач должно эффективно упорядочиваться в соответствии с установленной дисциплиной диспетчеризации, определенной при производстве системы, и реальным временем приема информации из внешней среды;
- алгоритмы и программы функциональных задач системы могут различаться по длительности решения и важности для пользователей, должны включать и использовать текущее и расчетное реальное время результатов решения и исходной информации;
- для эффективного использования ограниченных ресурсов производительности и оперативной памяти ЭВМ целесообразно применять приоритеты и прерывания исполнения программ в соответствии с выбранными дисциплинами решения различных функциональных задач;
- информация и сообщения для потребителей и внешней среды могут выдаваться асинхронно, в соответствии с установленной дисциплиной и содержать значения реального времени, которому соответствуют данные в сообщениях.

Требования к характеристикам качества сложных программных комплексов

Понятие ***корректной (правильной) программы*** может рассматриваться статически вне ее исполнения во времени. Корректность программы не определена вне области изменения исходных данных, ***заданных требованиями спецификации***, и не зависит от динамики функционирования программы в реальном времени. Степень некорректности программ определяется вероятностью попадания реальных исходных данных в пространство значений, которое задано требованиями спецификации и технического задания, однако не было проверено при тестировании и испытаниях. Значения этого показателя зависят от функциональной корректности применяемых компонентов и могут рассматриваться в зависимости от методов их достижения и оценивания: детерминировано, стохастически и в реальном времени.

Надежная программа, прежде всего, должна обеспечивать достаточно низкую вероятность отказа в процессе функционирования в реальном времени. Быстрое реагирование на искажения программ, данных или вычислительного процесса и восстановление работоспособности за время меньшее, чем порог между сбоем и отказом, обеспечивают высокую надежность программ. При этом не корректная программа может функционировать абсолютно надежно. В реальных условиях по различным причинам исходные данные могут попадать в пространство значений, вызывающих сбои, не проверенные при испытаниях, а также не заданные требованиями спецификации и технического задания. Если в этих ситуациях происходит достаточно быстрое восстановление, такое что 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 и существенно отличается от понятия корректности программ.

При оценке реализации требований надежности регистрируются только такие искажения в процессе динамического исполнения программ, которые приводят к потере работоспособности продукта. Работоспособность комплекса можно гарантировать при конкретных исходных данных, которые использовались при тестировании и испытаниях. Реальные исходные данные могут иметь значения, отличающиеся от заданных требованиями и от используемых при применении программ. При таких исходных данных функционирование программ трудно предсказать заранее, и весьма вероятны различные аномалии, завершающиеся отказами.

Непредсказуемость вида, места и времени проявления дефектов в процессе эксплуатации приводит к необходимости создания специальных, дополнительных систем оперативной 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 комплекса программ до устранения их первичных источников. Для этого в комплекс программ должна вводиться временная, программная и информационная избыточность, осуществляющая оперативное обнаружение дефектов функционирования, их иденти-

фикацию и автоматическое восстановление (рестарт) нормального функционирования. Эффективность такой защиты зависит от используемых методов, координации их применения и выделяемых вычислительных ресурсов на их реализацию.

Основным ***принципом классификации сбоев и отказов*** в программных продуктах при отсутствии их физического разрушения является разделение по временному показателю длительности восстановления после любого искажения программ, данных или вычислительного процесса, регистрируемого как нарушение работоспособности. При длительности восстановления, меньшей заданного порога, дефекты и аномалии при функционировании программ следует относить к ***сбоям***, а при восстановлении, превышающем по длительности пороговое значение, происходящее искажение соответствует ***отказу***. Классификация программных сбоев и отказов по длительности восстановления приводит к необходимости анализа динамических характеристик объектов, являющихся потребителями данных, обработанных комплексом программ, а также временных характеристик функционирования программ. Временная зона перерыва нормальной выдачи информации и потери работоспособности, которую следует рассматривать как зону сбоя, тем шире, чем более инертный объект находится под воздействием сообщений, подготовленных программным комплексом.

Требования к надежности программных продуктов в значительной степени адекватны аналогичным характеристикам, принятым для других технических систем. Наиболее широко используется ***критерий длительности наработки на отказ***. Для определения этой величины измеряется время работоспособного состояния системы между последовательными отказами или началами нормального функционирования системы после них. Вероятностные характеристики этой величины в нескольких формах используются как разновидности критериев надежности.

Основным ***требованием к процессу восстановления*** является ***длительность восстановления*** и ее вероятностные характеристики. Этот критерий учитывает возможность многократных отказов и восстановлений. Обобщение характеристик отказов и восстановлений производится в критерии ***коэффициент готовности***. Этот показатель отражает вероятность иметь восстанавливаемую систему в работоспособном состоянии в произвольный момент времени. Значение

коэффициента готовности соответствует доле времени полезной работы комплекса программ и системы на достаточно большом интервале, содержащем отказы и восстановления.

Наработка на отказ учитывает ситуации потери работоспособности, когда длительность восстановления достаточно велика и превышает пороговое значение времени, разделяющее события сбоя и отказа. При этом большое значение имеют средства оперативного контроля и восстановления (рестарта). Качество проведенной отладки комплекса программ более полно отражает значение длительности между потерями работоспособности программ – ***наработка на отказовую ситуацию или устойчивость***, независимо от того, насколько быстро произошло восстановление. Средства оперативного контроля и восстановления не влияют на наработку на отказовую ситуацию, однако могут значительно улучшать показатели надежности программ. Поэтому при оценке необходимого тестирования целесообразно измерять и контролировать наработку на отказовую ситуацию, а объем и длительность тестирования в ряде случаев устанавливать по наработке на отказ с учетом эффективности средств рестарта.

Непредусмотренные при проектировании ситуации и ошибки функционирования программ и данных могут быть потенциальными ***источниками катастроф*** при применении таких программных комплексов, ***влияющими на безопасность их функционирования и применения***. Наиболее полно функциональная безопасность комплексов программ характеризуется величиной ущерба, возможного при проявлении дестабилизирующих факторов и реализации конкретных ***угроз – рисков***, а также средним временем между проявлениями непредумышленных угроз, нарушающих надежность и безопасность. Однако описать и измерить в общем виде возможный ущерб при нарушении безопасности для критических комплексах программ разных классов практически невозможно. Поэтому реализации угроз целесообразно характеризовать интервалами времени между их проявлениями, нарушающими безопасность применения программ, или наработкой на отказы, отражающиеся на безопасности. Это ***сближает понятия и требования безопасности с показателями надежности*** комплексов программ. Различие состоит в том, что в показателях надежности учитываются все реализации отказов, а в характеристиках функциональной безопасности следует регистрировать только те ***случайные катастрофические отказы***, которые

отразились на безопасности. Статистически таких отказов может быть в несколько раз меньше, чем учитываемых в значениях надежности. Однако методы, влияющие факторы и реальные значения показателей надежности, могут служить ориентирами при оценке функциональной безопасности критических программных комплексов.

Оценка и утверждение ***целей функциональной безопасности*** требуется для демонстрации заказчику или пользователю, что установленные цели проекта адекватны проблеме его безопасности. Существуют цели и функции безопасности для комплекса программ и цели безопасности для среды. Рекомендуется сопоставлять эти цели безопасности с идентифицированными угрозами, которым они противостоят, и/или с политикой и предположениями, которым они должны соответствовать. Использование стандарта означает, что ***требования могут быть четко идентифицированы***, что они автономны, и применение каждого требования возможно и даст значимый результат при оценке качества, основанный на анализе соответствия комплекса программ этому конкретному требованию.

Требования к эффективности использования ресурсов ЭВМ программным комплексом в реальном времени

Для систем реального времени особое значение имеют ***требования эффективного использования*** программным продуктом ресурсов ЭВМ по производительности. Эффективность в стандарте **ISO 9126** отражена двумя динамическими характеристиками требований – временной эффективностью и используемостью ресурсов ЭВМ, которые рекомендуется описывать, в основном, количественными атрибутами, учитывающими динамику функционирования комплексов программ. Основные требования к характеристикам эффективности использования вычислительных ресурсов сосредоточены на наиболее критичных показателях производительности и длительности решения функциональных задач (см. рис. 1.5). При этом в контракте, техническом задании и спецификации требований должны быть зафиксированы и утверждены ***требуемые значения*** этих характеристик и их приоритетов. В стандарте **ISO 9126** эти характеристики качества комплексов программ рекомендуется отражать рядом атрибутов, каждый из которых следует оценивать для средних и наихудших сценариев функционирования комплекса программ.

Временная эффективность: свойства комплекса программ, отражающие требуемое время обработки заданий, ***время отклика*** из ЭВМ в систему и/или внешнюю среду после получения типового задания и начала решения требуемой функциональной задачи; а также производительность решения задач с учетом количества используемых вычислительных ресурсов (см. стандарт **ISO 14756)**. Временная эффективность программного комплекса определяется длительностью выполнения заданных функций и ожидания результатов системой или пользователями в средних и/или наихудших случаях с учетом приоритетов задач. ***Пропускная способность*** решения функциональных задач – производительность, число заданий, которое можно реализовать на данной ЭВМ в заданном интервале времени в зависимости от их содержания и числа действующих пользователей или воздействий из внешней среды.

Используемость ресурсов: степень загрузки доступных вычислительных ресурсов в течение заданного времени при выполнении функций комплекса программ в установленных условиях. Ресурсная экономичность отражается занятостью ресурсов центрального процессора, оперативной, внешней и виртуальной памяти, каналов ввода-вывода, терминалов и каналов сетей связи исполнением программ.

Потребность в производительности ЭВМ в процессе решения функциональных задач может значительно изменяться в зависимости от их свойств, а также от потока, состава и объема исходных данных. Степень использования ограниченной производительности ЭВМ в некоторых пределах не влияет на качество решения функциональных задач комплексом программ. При излишне высокой интенсивности поступления исходных данных может ***нарушаться временной баланс*** между длительностью решения полной совокупности задач комплексом программ в реальном времени, и производительностью ЭВМ при решении этих задач. Для формирования требований к программному продукту при подготовке технического задания и спецификаций ***следует согласовывать с заказчиком динамическую модель и характеристики внешней среды***, в которой будет применяться комплекс программ, а также динамику приема и передачи данных пользователям или системе. Оценивание величины производительности рекомендуется для определения: загрузки операторов-пользователей, пропускной способности по числу задач в единицу времени, временной шкалы событий обработки заданий и данных в системе. Эти ре-

зультаты предлагается сравнивать с требованиями заказчика и пользователей для оценивания рабочих нагрузок и достаточности производительности программного комплекса в конкретной системе и внешней среде.

Эти условия следует детализировать до уровня, позволяющего однозначно формализовать ***требования к допустимым значениям интенсивности решения функциональных задач***:

- в среднем, нормальном режиме работы программного комплекса с наибольшим качеством функциональной пригодности;
- в режиме предельной загрузки, реализующейся с определенной вероятностью, с допустимым снижением функциональной пригодности и некоторых конструктивных характеристик качества комплекса программ;
- в режиме кратковременной, аварийной перегрузки, способной критически отражаться на функциональной пригодности, надежности и безопасности применения программного комплекса.

Наиболее сложным является ***задание требований эффективности динамического использования производительности ЭВМ в реальном времени***. При этом должна быть определена зависимость качества решения функциональных задач от интенсивности поступающей информации различных типов. Основная задача состоит в определении вероятностей и ***рисков***, с которыми может нарушаться соответствие между потребностями в производительности для решения всей требуемой совокупности задач и реальными возможностями ЭВМ и других компонентов системы. Если эта вероятность невелика, и можно считать допустимым эпизодическое снижение качества за счет получающихся задержек и пропусков в обработке сообщений или заданий, то делается вывод о соответствии производительности ЭВМ функциям данного программного комплекса. Проблема обусловлена сложностью оптимального распределения в динамике ограниченных ресурсов ЭВМ (особенно производительности) по многим функциональным задачам, необходимостью проектирования программ с учетом этих ограничений и возможными переделками программ для того, чтобы соблюсти ресурсные ограничения.

Проверка корректности функциональных требований к сложным комплексам программ

В самом начале и в процессе развития проекта необходимо проверять ***качество и корректность требований***. Преимущества такого подхода состоят в том, что сводятся к минимуму дорогие переделки за счет уменьшения числа дефектов требований, которые можно обнаружить или предотвратить на ранних этапах жизненного цикла проекта или его версии. Определение требований напрямую связано с процедурами проверки и утверждения (аттестации), как это сформулировано в стандарте **ISO12207**. Принято считать, что требования описаны не полностью, если для них не заданы правила ***проверки и аттестации***, то есть, не определены способы контроля корректности и утверждения. Процедуры проверки требований являются отправной точкой для специалистов по качеству, непосредственно отвечающих за соответствие получаемого программного продукта предъявляемым к нему требованиям.

Требования должны быть ***верифицируемыми***. Требования, которые не могут быть проверены и аттестованы (утверждены) – это всего лишь ***не обязательные пожелания***. Именно так они буду восприниматься разработчиками, даже в случае их высокой значимости для заказчика и пользователей. Если описанное требование не сопровождается процедурами проверки, то в большинстве случаев говорят о недостаточной детализации или неполном описании требования и, соответственно, спецификация требований должна быть отправлена на доработку, и если необходимо, должны быть предприняты дополнительные усилия, направленные на конкретизацию требований. Процедура анализа требований считается выполненной только тогда, когда все требования, включенные в спецификацию, обладают методами оценки соответствия им создаваемого программного продукта.

В стандарте **IEEE 830** рекомендуется набор критериев качества формулировки требований к комплексам программ, который включает:

- корректность – отсутствуют дефекты и ошибки в формулировках требований к комплексу программ;
- недвусмысленность – каждое требование должно быть однозначно и не допускать различного понимания и толкования специалистами;

- полнота – состав и содержание требований должны быть достаточны для разработки и применения корректного комплекса программ и компонентов;
- непротиворечивость – между разными требованиями к компонентам и комплексу программ отсутствуют конфликты и противоречия;
- упорядоченность по важности и стабильности требований должна позволять определять их приоритеты и сохранять их при производстве и развитии комплекса программ;
- поддающееся проверке – должны допускать достоверный контроль каждого требования при применении, реализации и измерении;
- модифицируемость – каждое требование допускает возможность его простого и согласованного изменения и развития при производстве комплекса программ;
- трассируемость – требование имеет однозначный идентификатор и возможность детализации и перехода к подчиненным требованиям к комплексу программ;
- понимаемость – заказчики, менеджеры и специалисты-разработчики одинаково понимают и трактуют все требования к комплексу программ.

Для каждого функционального и количественного требования необходимо ***формировать меру качества***, которая позволит разбить все решения на два класса: решения, по которым достижимо соглашение специалистов о том, что они удовлетворяют требованию к комплексу программ, и решения, по которым делается вывод, что они не удовлетворяют требованию. Если определить меру качества для каждого требования, то каждое решение, удовлетворяющее этой мере, является приемлемым для заказчика и/или пользователя. Прежде всего, надо найти свойство требования, которое предоставит шкалу для его измерения. Например, если решено измерять время отклика системы на требование выполнения некоторой функции, то заинтересованные стороны рассматривают функционирование системы и приходят к заключению, что система не будет удовлетворять требованию, если время отклика на запрос пользователя составит больше заданного. Таким образом, время отклика становятся критерием качества для данного требования. Если решение вынуждает пользователя ожидать отклика системы больше заданного, следовательно, оно не удовлетворяет требованию.

Определения критериев качества требований позволяют выявить и прояснить ***нечеткие, неоднозначные требования***. Для них в одних случаях можно определить согласованный критерий качества, а в других случаях получается критерий, по которому не достигнуто соглашение с заказчиком. Такое неоднозначное требование целесообразно заменить несколькими требованиями, каждое из которых будет иметь свой собственный критерий качества. Не каждое требование может иметь критерий качества, который можно использовать для проверки того, удовлетворяет ли какое-либо решение требованию. Добавив разъяснения в критерий качества для каждого требования, можно сделать их осязаемыми и понятными. Это первый шаг по определению критериев для измерения качества решений. Подспорьем в поиске неосознаваемых и некорректных требований является создание моделей и/или прототипов, с помощью которых можно показать заинтересованным сторонам различные точки зрения на некоторые требования.

Когда формулируются требования помимо тех, которые уместны для конкретного проекта, есть вероятность, прихватить и несущественные требования, которые часто появляются в результате ***непонимания заинтересованной стороной целей проекта***. Специалисты, особенно если они уже имели неудачный опыт разработки систем, стремятся добавить эти требования ***«на всякий случай»***. Другая причина включения несущественных требований – личные пристрастия. Если заинтересованной стороне нужно что-то особенное, она полагает, что это является требованием, даже если это несущественно для проекта системы. Уровень конкретизации и неоднозначности требований взаимосвязаны и руководителям-менеджерам проекта необходимо стремиться установить баланс - ***«золотую середину»*** между ними. Для этого можно использовать рекомендации стандартов, шаблоны типовых документов или формальные математические методы описания и схемы представления требований.

Чтобы протестировать формулировки ***требований на значимость***, следует сопоставить их и сформулированные цели разработки программного продукта:

- способствует ли требование достижению целей проекта;
- если исключить требование, помешает ли это достижению целей;

- существуют ли другие избыточные требования, которые зависят от данного и адекватны им.

Когда представленные ***решения для требований ошибочны***, часто пропускаются необходимые требования. Также возможное решение не столь хорошее, каким оно могло бы быть, поскольку проектировщик не был свободен в выборе возможных способов для удовлетворения требований. Не всегда просто представить разницу между требованием и решением. Иногда в состав требований попадает часть технологии, и заказчик заявляет, что новый проект обязан использовать данную технологию. Если требование включает в себя часть технологического решения, то при условии, что указанное технологическое решение не является реальным ограничением, это требование на самом деле является решением. Имея представление о значении, которое придают заинтересованные стороны каждому требованию, можно расставить приоритеты требований к функциям и характеристикам программного продукта. Используя эти знания, можно выстроить приоритеты и компромиссные решения при разработке требований.

Необходимо уметь проверять, что разрабатываемый комплекс программ удовлетворяет каждому из зафиксированных и утвержденных требований. Нужно выделять каждое такое требование, чтобы ***проследить его развитие*** в ходе подробного анализа, проектирования и, наконец, реализации программ. Каждый этап разработки комплекса программ формирует, уточняет и реорганизует требования, чтобы сделать их как можно ближе к назначению нового программного продукта. Во избежание потерь и искажения требований нужно уметь преобразовывать исходные требования в решения для того, чтобы использовать это при тестировании.

Каждое требование должно иметь ***уникальный идентификатор***. Наилучший способ – присвоить каждому требованию номер. Каждое требование должно отражать ***отдельно распознаваемую, измеряемую сущность***. Необходимо определять и фиксировать связи между требованиями и влияние одних требований на другие. В проектах сложных комплексов программ нужно применять способ работы с большим числом требований и сложными связями между ними. Следует учитывать, что наряду с существованием некоторого числа требований, связанных с одним событием и/или сценарием использования, любое требование может быть связано с другими событиями и/или

сценариями использования. Событие и/или сценарий использования требований предоставляют некоторое количество небольших минимально взаимосвязанных систем. С помощью конфигурационного управления требованиями можно прослеживать, с какими событиями и/или сценариями использования какие решения связаны. Если в требование вносится изменение, необходимо определить все части системы, на которые влияет это изменение.

Спецификация требований должна содержать все ***требования, которым обязан удовлетворять программный продукт***. В спецификации необходимо объективно определить все, что он должен делать, а также те условия внешней среды, при которых он должен применяться и функционировать. Наиболее ответственный аспект формирования требований – общение со специалистами, которые вносят и изменяют требования. При наличии согласованного способа формирования требований все заинтересованные стороны могут принимать участие в процессе определения требований. Как только сформулировано хотя бы одно требование, можно приступать к его тестированию на корректность и задавать заинтересованным сторонам подробные вопросы для уточнения и конкретизации тестов. Можно применять различные тесты для ***проверки того, что требование существенно*** и что все ответственные участники проекта понимают его смысл одинаково. Целесообразно заказчику определять относительную значимость, приоритеты требований, задавать критерий качества для каждого требования и использовать этот критерий для тестирования корректности возможных решений.

Лекция 1.4

ТРЕБОВАНИЯ К ПОВТОРНОМУ ИСПОЛЬЗОВАНИЮ ГОТОВЫХ КОМПОНЕНТОВ ПРИ ПРОИЗВОДСТВЕ ПРОГРАММНЫХ КОМПЛЕКСОВ

Повторное использование компонентов в комплексах программ

Требования заказчиков и пользователей по совершенствованию и снижению затрат на информатизацию объектов и процессов отразились на формировании основных ***целей и требований создания и повторного применения мобильных программных компонентов и модулей***, которые состоят в следующем:

- обеспечение сохранения инвестиций, вложенных в реализованные и апробированные программные комплексы и компоненты, в процессе развития, модификации и появления новых требований к ним, а также при совершенствовании архитектур и возрастании ресурсов и функций аппаратных и операционных платформ;
- снижение трудоемкости, стоимости и длительности непосредственной разработки сложных программных продуктов;
- обеспечение возможности эффективного по экономическим показателям и качеству переноса апробированных программных компонентов и комплексов с минимальными изменениями на различные операционные и аппаратные платформы;
- экономная реализация совместной работы и расширения функций программных продуктов во взаимодействии с другими комплексами программ при решении крупной единой целевой задачи на различных локальных и распределенных платформах;
- обеспечение взаимодействия пользователям с программными продуктами в унифицированном стиле, облегчающем им переход к использованию новых или с расширенными функциями системам.

На обеспечение мобильности (переносимости) компонентов и комплексов программ и данных направлена значительная часть методов и средств ***современной программной инженерии***. Четкое разделение результатов работ, выполняемых на каждой стадии жизненного цикла комплекса программ, и определенные условия переходов между этапами жизненного цикла позволяют выделить следующие ***уровни повторного использования программ***:

- модели предметной области и спецификаций требований, возможно, реализуемые разными способами на этапе проектирования комплекса программ;
- проектные спецификации требований на этапе разработки комплекса программ;
- исходные тексты программ на языках программирования, применявшиеся при разработке повторно используемых программных компонентов;
- тесты проверки функционирования крупных компонентов, тесты проверки соответствия повторно используемых программ стандартизированным интерфейсам, комплексных тестов.

Требования повторного использования текстов и тестов компонентов и комплексов программ охватывают (рис. 1.6):

- возможность встраивания готового программного компонента в создаваемую новую систему при условии, что поставщик гарантирует его функционирование на выбранной платформе;
- перенос компонентов и/или комплексов программ и данных с платформ, в среде которых они были ранее реализованы, на выбранную для системы новую операционную и/или аппаратную платформу;
- обеспечение доступа к информационным ресурсам других распределенных систем и сетей.

Для этого следует создавать методологию и технологию, а также применять стандарты, поддерживающие требования и разработку повторно используемых и/или переносимых (мобильных) программ и данных (см. стандарт **ISO 14764**). Основные особенности требований к компонентам и комплексам программ для повторного использования в системах ***определяют две группы задач***:

- ***структурирование*** программ и данных на стадии проектирования систем, предполагающее последовательную декомпозицию заданных функций системы, что позволяет выделять программные

компоненты и модули, которые могут быть применены повторно как готовые, и описание их взаимодействия с другими компонентами;

Требования к повторному использованию готовых компонентов при производстве программных комплексов должны включать:

- повторное использование модулей, компонентов в комплексах программ:
  - цели и требования создания и повторного применения мобильных программных компонентов и модулей;
  - оценки рентабельности применения требований мобильности к программным компонентам;
  - затраты на процессы переноса программ и данных на иные платформы;
  - преимущества повторного использования программных компонентов;
- требования к подготовке модулей и компонентов для повторного использования в программных комплексах;
  - формирование повторно используемых компонентов при производстве комплексов программ;
  - недостатки и проблемы производства комплексов программ с ПИК;
  - подготовку возможности многократного использования компонентов в различном операционном и внешнем окружении;
- оценки эффективности повторного использования программных модулей и компонентов при производстве программных комплексов:
  - факторы, определяющие эффективность производства программного комплекса на базе готовых программных компонентов;
  - затраты на подготовку и формирование повторно используемых компонентов;
  - затраты на применение готовых компонентов;
  - изменения трудоемкости и длительности создания комплексов программ из ПИК;
  - особенности повторного использования крупных «коммерческих» программных компонентов;
- применение стандартов интерфейсов Открытых систем при производстве компонентов для повторного использования в программных комплексах:
  - обеспечение взаимодействия с иными компонентами, программными продуктами и внешними системами;
  - стандартизация обеспечения переносимости программных компонентов на уровне исходных текстов.

Рис. 1.6

- ***сборку или интеграцию*** готовых компонентов, а также комплексное, квалификационное тестирование программного продукта в целом.

Процессы переноса программ и данных на иные платформы, выбор методов обеспечения мобильности комплексов программ и характеристики используемых ресурсов для их реализации, прежде всего, зависят от параметров компонентов, предполагаемых для переноса. Ресурсы требуются в той или иной степени на ***двух фазах реализации требований*** переноса комплексов и компонентов программ:

- при создании потенциально переносимых компонентов и комплексов программ, когда требования эффективной мобильности предусматриваются и реализуются при их разработке и определяются возможные платформы и области повторного применения таких программ и данных;
- при непосредственной реализации требований переноса компонентов и/или программных комплексов, в различной степени подготовленных для переноса на иные платформы и/или для повторного использования на той же платформе.

Выбор и выделение компонентов и комплексов для повторного использования и/или переноса на другие аппаратные и операционные платформы зависит, прежде всего, от их размера и от кратности возможного применения. При разработке комплексов небольшого масштаба (порядка нескольких тысяч строк исходного текста) поиск и подбор готовых компонентов для их применения в новом комплексе чаще всего оказываются не рентабельными. Таким образом, существует некоторый диапазон малых размеров комплексов программ («докритическая масса»), для которых нецелесообразно искать готовые компоненты и применять «***сборочное программирование»*** из них, для которых ***нецелесообразно применять*** ранее созданные программы и массивы данных. По этому параметру можно выделить ***методологии переноса***:

- комплексов программ и компонентов, а также операционной среды в целом, решающих все функциональные задачи определенной сложной системы и полностью сохраняющих свою структуру на новой аппаратной платформе;
- достаточно автономных, крупных комплексов и массивов информации баз данных, решающих функциональные задачи во взаи-

модействии с имеющимися на новой аппаратной платформе операционными средствами;

- отдельных небольших функциональных компонентов программ и массивов данных для расширения и совершенствования функций ранее реализованных задач на той же аппаратной и операционной платформах.

В каждом конкретном случае ***необходима оценка рентабельности применения требований мобильности*** к компонентам с учетом ряда факторов, характеризующих комплекс программ и данных и их среду, по сравнению с полной разработкой аналогичных программных продуктов. Для достижения перечисленных целей и обеспечения мобильности требуются различные ресурсы при их реализации. Потребность в конкретных ресурсах и рентабельность их использования зависит от ряда параметров, которые образуют широкий спектр ситуаций для анализа и применения свойства мобильности программ и данных. Такими ***ресурсами являются***:

- трудовые затраты специалистов и время на создание требований к дополнительным интерфейсным компонентам в программах и данных, обеспечивающим их эффективную мобильность на определенные типы платформ, например, в соответствии с концепцией и стандартами Открытых систем;
- дополнительные ресурсы памяти и производительности вычислительных средств, необходимые для реализации и функционирования компонентов программ, обеспечивающих их высокую мобильность, например, для реализации стандартизированных интерфейсов с внешней и внутренней средой;
- трудовые затраты специалистов и время на создание, приобретение и эксплуатацию инструментальных средств, автоматизирующих разработку и сопровождение мобильных комплексов программ.

При переносе программного продукта свойства системы всегда несколько изменяются, что следует учитывать при анализе целесообразности переноса, а также могут быть необходимы его отладка, испытания и сертификация в новой среде. Следует также учитывать, что любой ***перенос связан с затратами***, которые чаще всего требуются для:

- системного анализа рентабельности требований переноса на иную или ту же платформу и оценки технико-экономических показателей этого процесса;

- реализации самого процесса переноса и интеграции компонентов и/или программного продукта с операционной и внешней средой на новой аппаратной платформе или в существующей среде;
- квалификационного тестирования, испытаний и комплексной проверки функционирования версии программного продукта в новом окружении или на новой платформе;
- сертификации перенесенных на новую платформу продуктов и функционирующих в иной операционной и внешней среде;
- корректировки или дополнения эксплуатационной и технологической документации на перенесенный программный продукт.

Два последних вида работ могут выполняться в процессе создания мобильных комплексов программ и отсутствовать при непосредственной реализации переноса. Однако и в этом случае ***следует избегать излишнего оптимизма при оценке затрат на перенос***, так как при создании мобильных продуктов трудно предусмотреть все возможные особенности различных платформ и внешней среды, для которых декларируется мобильность конкретных программных средств. Эти особенности и возможное расширение окружающих прикладных программ и данных могут преподносить неприятные сюрпризы нестыковки, для ликвидации которых потребуются дополнительные затраты.

Проектирование систем ***с повторно используемыми компонентами (ПИК)*** стало ***особенно рентабельным для сложных программных продуктов***, содержащих сотни или тысячи модулей, и с большими объемами обрабатываемой информации. ***Кратность применения готовых компонентов*** также значительно влияет на эффективность их переноса. Особенно важно тщательно формулировать требования, осуществлять унификацию интерфейсов и оформление документации для тех компонентов, которые в перспективе будут использоваться многократно различными специалистами в различных вариантах на той же или на различных платформах.

Для обеспечения переноса текстов программных компонентов необходимо при их первичной разработке ***подготовить потенциальную возможность их последующего многократного использования*** в различном операционном и внешнем окружении. Для этого должна быть унифицирована технология разработки модулей и групп программ, унифицированы структуры межмодульных связей и технология комплексирования, стандартизирована система идентификации

и специфицирования программ и данных, а также дисциплина испытаний и документирования компонентов и комплексов программ. Программные компоненты должны быть подготовлены для ***отчуждения*** от их первичных разработчиков и от комплексов программ, в составе которых они первично отлаживались, испытывались и применялись. Таким образом, возникает необходимость проведения ряда ***общесистемных и организационных работ***, а также анализа некоторых дополнительных затрат на обеспечение возможности эффективного переноса программ и данных в различную внешнюю среду.

При этом проявляются следующие ***преимущества повторного использования программных компонентов:***

- ускорение разработки – повторное использование компонентов ускоряет создание систем, так как сокращается время на их разработку и тестирование;
- повышение надежности – компоненты, повторно используемые в других системах, оказываются значительно надежнее новых компонентов, они протестированы и проверены в разных условиях работы, ошибки, допущенные при их проектировании и реализации, обнаружены и устранены при первом их применении;
- уменьшение проектных рисков – для уже существующих компонентов можно более точно прогнозировать расходы, связанные с их повторным использованием, такой прогноз – важный фактор организации комплекса программ;
- эффективное использование специалистов – часть специалистов, выполняющих одинаковую работу в разных проектах, может заниматься разработкой компонентов для их дальнейшего повторного использования, эффективно применяя накопленные ранее знания и опыт;
- стандарты интерфейса компонентов, внешней среды и пользователя можно реализовать в виде набора стандартных компонентов, что повышает надежность систем, так как, работая со знакомым интерфейсом, пользователи совершают меньше ошибок.

Для успешного проектирования и разработки комплексов с повторным использованием компонентов должны выполняться ***следующие основные условия:***

- возможность поиска необходимых программных компонентов – должен быть каталог документированных апробированных

компонентов, предназначенных для повторного использования, который обеспечивал бы быстрый поиск нужных компонентов;

- необходимо удостовериться, что поведение компонентов предсказуемо и надежно, все компоненты, представленные в каталоге, должны быть сертифицированы, чтобы подтвердить соответствие определенным стандартам качества;
- на каждый компонент должна быть соответствующая документация, чтобы можно было получить нужную информацию о компоненте и адаптировать его к новому приложению, и информация о том, где и как используется данный компонент.

Требования к подготовке компонентов для повторного использования в программных комплексах

Во многих случаях компоненты, комплексы программ и информация баз данных создавались и разрабатываются до сих пор ***без ориентации на последующее повторное использование и/или на перенос на иную аппаратную и операционную платформы***. Это обусловлено отсутствием стремления у разработчиков доводить программы и их документацию до уровня завершенных поставляемых компонентов и продуктов**.** Многие международные стандарты, регламентирующие создание переносимых приложений, не доступны отечественным специалистам из-за низкой системной и программистской культуры и отсутствия стремления освоить и использовать современные стандарты программной инженерии. Огромный парк ***унаследованных*** программных продуктов и их компонентов разработан под конкретные проекты и платформы и является потенциально мобильным только в пределах, расширяющихся по параметрам и ресурсам комплексов определенной архитектуры.

Повторное использование компонентов должно быть ***систематическим, плановым и включенным во все организационные программы предприятия-разработчика***. Хотя такой подход может привести к значительному увеличению количества повторно используемых компонентов. Перед началом этапа проектирования разработчики должны выполнять поиск компонентов, подходящих для повторного использования. Системная архитектура строится на основе уже имеющихся (готовых) компонентов. Требования к системе изменяются с учетом имеющихся компонентов, выбранных для повторного использования. Такой подход предполагает определенные компромиссы

в реализации требований. Хотя такая система может оказаться менее эффективной, чем система, разработанная без ПИК, этот недостаток компенсируется более низкой стоимостью разработки, более высокими темпами создания системы и ее повышенной надежностью.

Унификация всегда требует некоторых ресурсов, которые в данном случае выражается в дополнительной трудоемкости создания ПИК программ и данных, а также в увеличении необходимой памяти и производительности ЭВМ для их реализации. Сохранение и развитие довольно широкого спектра архитектур ЭВМ естественно привело к повторному использованию компонентов не только на однотипных платформах, но и к производству программных средств и баз данных, переносимых на различные аппаратные и операционные платформы. Таким образом, сформировались ***две технологические проблемы***:

- ***создание*** программных компонентов и баз данных, которые рентабельно повторно применять в различных проектах и/или переносить на различные операционные и аппаратные платформы;
- ***применение*** повторного использования и/или переноса компонентов программ и баз данных для создания из них новых программных продуктов на различных платформах.

Формирование повторно используемых компонентов при производстве комплексов программ может осуществляться использованием стандартов и руководств, обязательных для оформления и производства основной массы программных компонентов (80 – 90%) с возможностью их применения как повторяющихся в определенном семействе комплексов программ, однако допускающем однократное применение (10 – 20%) уникальных компонентов с произвольным оформлением. Особенно полное тестирование и оформление документации целесообразно проводить для тех компонентов, которые в перспективе будут использоваться многократно различными специалистами и в разных проектах. При анализе характеристик производства конкретных комплексов трудно предвидеть кратность повторного использования определенных компонентов в будущих проектах. Поэтому обычно затраты на их производство рассматриваются отдельно или включаются в экономические характеристики первой версии программного продукта, в которой они применены.

Важная особенность программных компонентов, пригодных для повторного использования, это целесообразность ***отделения специ-***

фикации требований и функций от реализации структуры компонента. Спецификация компонента должна содержать функции компонента, и как он поведет себя, когда его возможности используются другими компонентами. Обладая этой спецификацией можно сосредоточиться на общем решении проблемы, не углубляясь в то, как реализованы функции компонента. В общем случае, могут существовать более одной версии реализации спецификации компонента – каждая со своими отличиями в определенных аспектах, таких как платформа, производительность или стоимость.

Возможности повторного применения спецификации компонента реализуются одной или несколькими ***спецификациями интерфейсов***, каждая из которых описывает взаимосвязанную группу возможностей взаимодействия. Для программного компонента спецификации интерфейса представленные в спецификациях компонента должны быть доступны во время применения и потенциально во время исполнения компонента. Чтобы гарантировать эффективную работу этих процессов и обеспечить возможность использования четко сформированных интерфейсных спецификаций во время исполнения, очень важен корректно определенный язык спецификаций интерфейсов.

Разработка для повторного использования должна быть процессом, основанным на опыте и знаниях о проблемах повторного использования создания обобщенных компонентов, которые можно адаптировать для разных вариантов их использования. Программный компонент, предназначенный для повторного использования, имеет ряд особенностей, он должен отражать стабильные абстракции предметной области, т.е. фундаментальные понятия области приложения, которые меняются медленно. Должен скрывать способ представления своего состояния и предоставлять операции, которые позволяют обновлять состояния и получать к нему доступ, быть максимально независимым.

Компонент должен быть настолько ***автономным***, чтобы не нуждаться в других компонентах. В действительности такое выполнимо только для простых компонентов, более сложные всегда зависят от других компонентов. Все исключительные ситуации должны быть частью интерфейса компонента. Компоненты не должны сами обрабатывать исключения, так как в разных приложениях существуют разные требования для обработки исключительных ситуаций. Лучше

определить те исключения, которые необходимо обрабатывать, и объявить их как часть интерфейса компонента.

В большинстве существующих систем имеются большие сегменты кода, которые реализуют ***абстракции предметной области***, однако их нельзя непосредственно использовать как ПИК. Причина в несоответствии программного кода модели четко определенному интерфейсу запросов и поставщиков сервисов. Чтобы повторно использовать такие компоненты, как правило, необходимо построить упаковщик – программу для создания оболочки и стандартизации внешних обращений. Упаковщик скрывает исходный код и предоставляет интерфейс для внешних компонентов, открывающий доступ к предоставляемым сервисам.

При создании компонентов, предназначенных для повторного использования, предполагается предоставление общего интерфейса с операциями, которые обеспечивают разные способы использования компонентов. Чтобы сделать компоненты практичными в использовании, требуется минимальный интерфейс, простой для понимания. С другой стороны, предполагаемая возможность повторного использования усложняет компоненты и потому уменьшает их понятность. Поэтому разработчики компонентов, предназначенных для повторного использования, должны прийти к некоторому компромиссу между обобщенностью и понятностью интерфейсов компонентов.

Производству комплексов программ с ПИК присущ ***ряд недостатков и проблем***, которые препятствуют запланированному сокращению расходов на разработку комплекса:

- недоступность исходного кода компонента может привести к увеличению расходов на сопровождение системы, так как повторно используемые системные элементы могут со временем оказаться не совместимыми с изменениями, производимыми в системе;
- некоторые разработчики предпочитают переписать компоненты, так как полагают, что смогут при этом их усовершенствовать. Кроме того, многие считают, что создание программ «с нуля» перспективнее повторного использования программ, написанных другими;
- заполнение библиотеки компонентов и ее сопровождение может стоить дорого, еще недостаточно хорошо продуманы методы классификации, каталогизации и извлечения информации о программных компонентах.

Увеличение затрат на производство повторно используемых компонентов должно компенсироваться сокращением затрат при производстве комплексов программ на их основе. Освоение методов и средств решения этих проблем позволяет ***качественно изменять экономические характеристики*** процессов производства сложных комплексов программ и резко повышать производительность труда специалистов при их разработке. Создание новых программных продуктов путем переноса их с других аппаратных и операционных платформ стало особенно ***актуальным для современных административных систем*** государственного и регионального управления, управления отраслями и предприятиями промышленности, а также банковскими системами и в социальной сфере.

Поставки и закупки вычислительной техники зачастую производятся с позиции только ее низкой цены без анализа возможности их взаимодействия в распределенной системе определенного функционального назначения и наличия соответствующих готовых ПИК. В ряде областей применения имеется широкий спектр типовых программных компонентов, которые могут быть использованы в различных сочетаниях для решения функциональных, проблемно-ориентированных задач. При их решении требования к эффективности использования ресурсов ЭВМ не столь жесткие, как в критических системах управления в реальном времени, и могут быть унифицированы интерфейсы между компонентами. Таким образом, выявилась необходимость, и сложились благоприятные условия для эффективного применения современных ***технологий производства программных продуктов путем переноса и повторного использования компонентов*** на различных аппаратных и операционных платформах .

При ***анализе экономических характеристик и рентабельности*** производства комплексов программ на базе готовых ПИК и/или путем их переноса с другой аппаратной и операционной платформы ***необходимо учитывать***:

- степень подобия архитектуры и соотношения основных параметров ресурсов аппаратных платформ, между которыми предполагается перенос и /или применение готовых компонентов;
- уровень унификации интерфейсов приложений с операционными платформами, между которыми предполагается перенос готовых программ и данных;

- наличие или отсутствие при разработке исходных программ и информации баз данных, ориентации на будущий перенос и/или повторное использование на других платформах и в других проектах.

Различия операционных систем, принципов организации вычислительного процесса и контроля функционирования, а также драйверов ввода-вывода могут практически полностью исключить эффективный автоматизированный перенос соответствующей части унаследованных готовых компонентов между разнотипными по архитектуре ЭВМ. Важнейшим фактором, влияющим на рентабельность переноса, является ***степень унификации интерфейсов программных продуктов с операционной и внешней средой и с пользователями***. Применение одних и тех же стандартизированных языков программирования и аттестованных компиляторов позволяет значительно сокращать затраты при переносе. Создание современных сложных программных продуктов целесообразно вести с использованием совокупности международных стандартов Открытых систем, часть которых обеспечивает мобильность программных средств.

Различие архитектуры и ресурсов ЭВМ, между которыми предполагается перенос программ и данных, могут быть фактором, определяющим экономическую эффективность повторного использования компонентов и программных комплексов. В состав этих параметров входят номенклатура и специфика реализации операций ЭВМ, структура и особенности системы адресации, структура и разрядность памяти, особенности реализации системы ввода-вывода и т.д. Значительные трудности может вызывать или делать практически не рентабельным перенос программ и данных существенное различие производительности и размеров памяти рассматриваемых ЭВМ. Особое значение для переносимости имеет специфика реализации взаимодействия с внешней средой и процедур ввода-вывода в ядре операционных систем сопоставляемых ЭВМ. Программы, реализующие эти функции, должны отражать специфику конкретной системы ввода-вывода и особенности источников и потребителей информации. Такая ориентация программ ввода-вывода может делать их практически не переносимыми и требовать выделения в автономные, заново разрабатываемые, автоматизировано непереносимые процедуры.

Оценки эффективности повторного использования компонентов при производстве программных комплексов

У разработчиков комплексов программ зачастую возникает проблема – проводить ли проектирование и производство ***нового*** сложного программного продукта по технологии полного жизненного цикла «с нуля», или попытаться найти прототипы и ***воспользоваться готовыми апробированными решениями и компонентами***. При этом критерием экономической эффективности использования готовых компонентов в первом приближении может быть относительное число компонентов или их доля от полной трудоемкости производства программного продукта, которая сокращается за счет повторного испльзования апробированных ПИК. Для этого, из полной трудоемкости разработки нового продукта может быть исключена часть, которая обусловлена затратами на первич-ное создание ПИК. Затраты на процессы проектирования комплекса программ при этом практически не изменяются, но сокращаются призводственные этапы программирования, тестирования и автономной отладки компонентов, а также этапы их сборки, и испытаний комплекса. На этих этапах уменьшению затрат может способствовать также накопленный опыт создания предшествующих версий.

Использование готовых программных компонентов позволяет сокращать или исключать из полных затрат на производство комплекса, ***затраты на программирование и тестирование компонентов и модулей***, которые зависят от относительного их числа и объема (доли ПИК) в составе общего числа компонентов в комплексе программ. Кроме того, большое число ПИК может отражаться на сокращении затрат на этапах предварительного и детального проектирования, также положительно влиять на уменьшение затрат на сборку, испытания и документирование программного продукта.

Для оценки эффективности «сборочного» программирования с применением ПИК необходимы характеристики трудоемкости разработки в зависимости от этапов жизненного цикла и влияния их доли на трудоемкость этапов. Влияние доли ПИК на другие этапы приходится оценивать, используя правдоподобные гипотезы и возможные варианты. Предположим, что каждый этап (из всех этапов жизненного цикла) разработки ***при полном цикле создания программного продукта с необходимым комплексом ПИК*** характеризуется

известной долей от полной трудоемкости производства программного продукта. На каждом этапе выделим долю труда, которую можно отнести к созданию и тестированию повторно используемых компонентов, а остальную часть будем считать относящейся к системному анализу, сборке и комплексированию компонентов. В первом приближении можно предположить, что общая величина затрат зависит от доли ПИК ***линейно*** в некоторых пределах, в зависимости от этапа производства. Для определения ***относительного изменения трудоемкости*** при создании продукта с помощью «сборочного» программирования необходимо учесть относительную роль, каждого этапа и долю снижения трудоемкости на соответствующем этапе за счет прменения ПИК.

Оценка изменения полной трудоемкости производства программного продукта за счет повторного использования компонентов зависит от доли готовых ПИК из полного состава программных компонентов в комплексе. Эти оценки могут быть проведены для простейшего случая, когда применяются все 100% готовых ПИК при некоторых частных предположениях о значениях затрат на последующих этапах производства. Предельный выигрыш в трудоемкости производства комплекса, собираемого из полного набора готовых ПИК, когда отсутствует необходимость разработки дополнительных программных компонентов, можно оценить на основе распределения затрат по этапам разработки следующим образом. Трудоемкость сокращается только ***за счет исключения этапов программирования и автономного тестирования новых программных компонентов и модулей*** в процессе производства. Трудоемкость этих этапов для программных комплексов реального времени (СРВ) составляет около 50% от полной трудоемкости при отсутствии ПИК. Таким образом, суммарная трудоемкость в данном варианте уменьшается для этого класса комплексов программ соответственно почти в 2 раза.

В процессе совершенствования «сборочного» программирования с применением ПИК возможно некоторое дополнительное снижение затрат при детальном проектировании и комплексной отладке. Однако итоговое снижение трудоемкости или ***повышение эквивалентной производительности труда*** при рассмотренных предположениях вряд ли превысит пятикратное. Такое снижение, по-видимому, максимальное при различии функциональных характеристик вновь создаваемого программного комплекса. При этом наличие

затрат на динамическую отладку комплекса программ для систем реального времени заметно снижает эффективность «сборочного» программирования по сравнению с эффективностью при производстве административных систем и является существенным ограничением снижения трудоемкости для программных комплексов этого класса. Применение «сборочного» программирования возможно и при глубокой функциональной преемственности последовательно разрабатываемых версий программных продуктов. В этих случаях доля затрат на системный анализ, детальное проектирование, комплексирование, тестирование и испытания может быть значительно меньше приведенных оценок. Вследствие этого эффективность «сборочного» программирования, соответственно, повысится.

На практике при создании нового программного продукта ***не всегда имеется полный набор готовых и пригодных для применения ПИК***. Тогда при сборке продукта может потребоваться доработка отдельных компонентов, их сопряжение в новых сочетаниях и создание новых программных компонентов для решения дополнительных задач. Поэтому целесообразно оценивать трудоемкость «сборочного» программирования с учетом частичных затрат на новые компоненты. В пределе при производстве программного продукта полностью из многократно применяемых готовых компонентов трудоемкость может сократиться в 3 – 5 раз. В промежуточных случаях, когда готовые компоненты используются частично, оценку изменения трудоемкости можно провести по степени сокращения затрат на программирование и автономное тестирование всех новых необходимых компонентов. В результате могут быть получены практически зависимости эффективности изменения трудоемкости от доли ПИК для комплексов реального времени.

Изменение длительности производства комплексов программ за счет применения ПИК относительно меньше, чем трудоемкости или производительности труда. Необходимость выполнения определенной совокупности производственных этапов и операций в заданной технологической последовательности остается более или менее постоянной при различных воздействиях на процесс производства. Исключением является применение ПИК, при котором значительно сокращаются этапы программирования и автономного тестирования модулей и программных компонентов, а также в той или иной степени длительность других этапов. В ***рассматриваемом ва-***

рианте при наличии полного набора ПИК можно исключить программирование и тестирование компонентов. В результате длительность разработки программных комплексов реального времени уменьшается приблизительно на 30% (в 1,5 раза).

Если необходимо вновь создать некоторую часть программных компонентов, ***полная длительность разработки может мало измениться по сравнению с разработкой без ПИК***. Это объясняется тем, что длительность программирования и автономного тестирования компонентов слабо зависит от того, какое количество новых компонентов предстоит создавать. Поэтому зависимость продолжительности производства, от доли ПИК оказывается нелинейной, и заметное сокращение длительности разработки проявляется только при создании программного комплекса ***практически полностью из готовых компонентов***.

Наиболее широко ПИК применяются в административных, финансовых, социальных системах, для которых разработано множество крупных пакетов функциональных программных продуктов и могут собираться проблемно-ориентированные комплексы программ для конкретных заказчиков. В ряде случаев оказывается необходимой разработка и/или замена только небольшой доли новых ПИК, в основном для обеспечения интерфейсов с пользователями. В таких системах значительную роль могут играть первичные капиталовложения в создание технологии и аппаратурное обеспечение средствами вычислительной техники и применения сложных ПИК. Эти виды затрат обычно сосредоточиваются в производстве первой версии программного продукта и в дальнейшем могут не учитываться.

Особенности повторного использования крупных «коммерческих» программных продуктов включают функциональность, которая намного шире функциональности более специализированных компонентов, и поэтому увеличивается потенциальный выигрыш, полученный от повторного использования. Некоторые типы «коммерческих» систем используются повторно на протяжении многих лет, например, базы данных. Благодаря функциональности, предлагаемой этими системами, сокращение финансовых и временных затрат может достичь величины, сравнимой с разработкой «с нуля». Более того, уменьшаются риски, так как крупные «коммерческие» системы уже существуют и разработчики могут увидеть, удовлетворяют ли они предъявляемым к ним требованиям.

В принципе, тестирование крупных модульных систем не отличается от использования других больших специализированных комплексов. Для этого необходимо изучить интерфейсы комплекса программ и использовать их для организации взаимодействия с другими системными компонентами, также необходимо разработать архитектуру, которая поддерживала бы крупные «коммерческие» системы при совместной работе. Однако тот факт, что такие программные продукты представляют собой крупные системы и часто продаются как отдельные автономные системы, вносит дополнительные проблемы в их тестирование. При интеграции таких систем могут возникнуть следующие проблемы:

- недостаточный контроль над функциональностью и производительностью крупных продуктов, хотя считается, что их интерфейсы известны, не исключена вероятность наличия скрытых операций, которые будут «пересекаться» с системными операциями;
- проблемы, связанные с организацией взаимодействия крупных комплексов, иногда сложно подобрать продукты для совместной работы, поскольку каждый продукт разрабатывался на основе различных предположений по поводу его использования;
- отсутствие контроля за модификацией таких крупных продуктов, их производители принимают решения по изменению своих комплексов под давлением рынка, новые версии могут обладать дополнительной функциональностью, неподдерживаемой предыдущими версиями;
- уровень поддержки, оказываемой производителями крупных продуктов, варьируется в широких пределах, поскольку эти системы распространяются свободно, поддержка производителей особенно важна в тех случаях, когда у разработчиков возникают проблемы, например, при тестировании, связанные с получением доступа к исходному коду и к подробной документации системы;
- несмотря на то, что производитель берет на себя обязательства по поддержке своих программных комплексов, изменение ситуации на рынке и экономических условий могут привести к тому, что ему станет трудно продолжать выполнение взятых обязательств по их сопровождению.

Применение стандартов интерфейсов Открытых систем при производстве компонентов для повторного использования в программных комплексах

Основными целями создания и применения концепции, методов и стандартов Открытых систем является повышение общей ***экономической эффективности производства*** и функционирования систем, а также логической и технической совместимости их компонентов для обеспечения повторного использования готовых программ и данных. Для достижения этих целей развиваются и применяются различные проблемно-ориентированные технологии и комплексы средств автоматизации производства комплексов программ, ***базирующиеся на повторном использовании апробированных программных компонентов, комплексов и данных***, их эффективном переносе на различные аппаратные и операционные платформы и согласованном взаимодействии в распределенных системах.

Задача сводится к ***максимально возможному повторному использованию разработанных и апробированных программных и информационных компонентов*** при расширении масштаба комплексов программ, изменении вычислительных аппаратных платформ, их операционных систем и процессов взаимодействия. Концепция и система стандартов Открытых систем должны обеспечивать возможность относительно простого и эффективного по трудоемкости переноса апробированных программных продуктов и информации баз данных на различные типы аппаратных платформ за счет стандартизации процессов и интерфейсов взаимодействия программ с операционными системами ЭВМ. Основной задачей является транспортировка и перенос функций и процедур обработки информации, а также содержания баз данных между различными платформами, осуществляющими их обработку. Подобные обмены функциями, процедурами и данными имеют неоперативный характер и могут осуществляться при проектировании вне реального масштаба времени. Проблемы состоят преимущественно в обеспечении сохранности апробированного функционального ядра текстов программ и информации баз данных при их переносе на иные аппаратные и операционные платформы для ***снижения трудоемкости и длительности производства сложных программных продуктов***.

Концепция Открытых систем – это совокупность международных стандартов, которая специфицирует интерфейсы, услуги и поддерживающие форматы для достижения взаимодействия и переносимости программных продуктов, данных и персонала, достаточные для того, чтобы обеспечивать:

- возможность переноса (***мобильность***) компонентов, комплексов программ и данных, разработанных должным образом, с минимальными изменениями на широкий набор аппаратных и операционных платформ;
- совместную работу (***интероперабельность***) компонентов с другими компонентами, программными комплексами и системами на локальных и удаленных платформах;
- взаимодействие с пользователями в стиле, облегчающем последним переход от системы к системе (***мобильность пользователей***).

Открытые системы делают доступным пользователям широкий круг программных продуктов и их компонентов, позволяют предприятиям сэкономить средства на документации и переподготовке специалистов, а кроме того делают их независимыми от конкретного поставщика программного и аппаратного продукта. Построение открытых систем из унифицированных по взаимодействию компонентов и комплексов программ стимулирует конкуренцию среди поставщиков как по соотношению цена/производительность, так и по функциональным возможностям. Профессионалы в области Открытых систем акцентируют усилия на поиске и создании гибкой, способной к наращиванию программной и информационной среды, которые ***базируются на трех направлениях стандартизации***:

- стандартизация аппаратных и операционных платформ;
- стандартизация методов и технологий обеспечения жизненного цикла программных компонентов и комплексов;
- стандартизация интерфейсов программных компонентов и комплексов между собой, с операционной и внешней средой.

Идеология Открытых систем существенно отражается на ряде направлений развития современных программных продуктов. Она базируется на строгом соблюдении совокупности протоколов и стандартов де-юре и де-факто. Программные и аппаратные компоненты должны отвечать ***двум важнейшим требованиям***: переносимости и возможности согласованной, совместной работы с другими,

возможно удаленными, компонентами. Это позволяет обеспечивать совместимость различных программных компонентов, комплексов, систем и средств передачи данных.

Технология Открытых систем для повторного использования компонентов и переноса комплексов программ на иные платформы наиболее сильно ***отражается на сокращении затрат ресурсов*** при создании крупных «коммерческих» административных систем обработки информации. При этом возможно снижение трудоемкости в 10 – 15 раз, длительности разработки в 5 – 8 раз и числа необходимых специалистов почти вдвое относительно значений при полной разработке без применения современной технологии переноса и ПИК. Эффективность переноса программных продуктов систем реального времени значительно ниже, чем в предыдущем случае, вследствие высокой доли затрат на перепрограммирование некоторых компонентов и интерфейсов, а также на постановку и освоение технологии на новой платформе. Для этого класса комплексов программ при переносе трудоемкость может уменьшаться в 4 – 5 раз, длительность в 2 – 3 раза, а необходимое число специалистов почти в два раза относительно полной разработки «с нуля» без применения ПИК.

Рядом зарубежных организаций и промышленных фирм под руководством **IEEE** с 1990 года ведется активная разработка и модернизация последовательных версий стандартов интерфейсов Открытых систем **POSIX** (Portable operating system interfaces). В результате подготовлена группа международных стандартов из четырех крупных частей **ISO 9945:1-4:2003 (IEEE 1003.1 – 2003)**. Цель документов – ***стандартизация обеспечения переносимости программных компонентов на уровне исходных текстов***. В них определены основные интерфейсы операционных систем и окружения, интерфейсы командного интерпретатора, а также программы общих утилит. ***Три отдельных крупных тома*** включают: базовые определения; системные интерфейсы; команды управления и сервисные программы (утилиты). Кроме того, имеется большой ***четвертый том*** общего обоснования выбранных решений системы **POSIX**. Важными свойствами разработанных программных интерфейсов являются целостность, модульность их построения и параметризуемость.

Стандарты открытых систем – **POSIX** регламентируют совокупность базовых, системных сервисов для ***обеспечения унифицированных интерфейсов прикладных программ***, специфицированных

для языка **Си**, командного языка и совокупности служебных программ. ***Основная цель –*** сделать программы ***переносимыми*** на уровне различных исходных языков. У каждого интерфейса компонентов программ существует вызывающая и вызываемая сторона, стандарты **POSIX** ориентированы преимущественно на формализацию вызывающей стороны. Мобильность приложений должна обеспечиваться благодаря применению большого числа стандартизированных системных интерфейсных сервисов и возможности динамического выяснения характеристик целевой платформы и подстройки под них интерфейсов компонентов и комплексов программ.

Кроме стандартов **POSIX**, разработано и применяется множество групп стандартов де-юре и де-факто для унификации архитектур и интерфейсов программных комплексов информационных систем различных классов. Разработку определенных стандартов стимулируют обычно коллективы высококвалифицированных в определенной прикладной сфере специалистов, которые привлекают другие организации. Одобрение и выпуск стандартов де-юре считается критерием высокого качества регламентируемых процессов.

* * *

При проектировании и производстве сложных программных продуктов целесообразно анализировать и выбирать подходящие интерфейсы группы Открытых стандартов. Их необходимо приобретать, строго соблюдать и контролировать при применении, что является некоторым ограничением производственных процессов и требует определенных затрат. Однако при этом одновременно ***повышаются экономические характеристики производства программных продуктов и их компонентов***, эффективность переноса на различные платформы, а главное - качество технологии и результатов производства.

Таким образом, для обеспечения эффективного управления разработкой программ и ***обеспечения их тестируемости*** необходимо ***стандартизировать и соблюдать ряд требований архитектурного построения и интерфейсов комплексов и компонентов программ***. Эти требования могут иметь особенности для проектов в различных проблемно-ориентированных областях. Однако их стандартизация обеспечивает значительный эффект в снижении трудоемкости и длительности последующей разработки и тестирования программных

продуктов и их версий. Частичная потеря гибкости архитектуры, некоторое возрастание ресурсов, необходимых для их реализации, обычно полностью компенсируются повышением управляемости и технико-экономических показателей процесса разработки, тестирования, а также качества программных продуктов.

Лекция 1.5

ТРЕБОВАНИЯ К ДОПУСТИМЫМ РИСКАМ И К ДОКУМЕНТИРОВАНИЮ ТРЕБОВАНИЙ К КОМПЛЕКСАМ ПРОГРАММ

Риски при формировании требований к характеристикам компонентов и программных комплексов

Рассматриваемые риски могут быть обусловлены нарушениями технологий или ограничениями при использовании ресурсов – бюджета, планов, коллектива специалистов, инструментальных средств, выделенных на разработку компонентов и комплекса программ. Результирующий ***ущерб*** в совокупности зависит от величины и вероятности проявления каждого негативного воздействия. Этот ущерб – риск характеризуется разнообразными метриками, зависящими от объектов анализа, и в некоторых случаях может измеряться прямыми материальными, информационными, функциональными потерями безопасности применяемых программ или систем. Одним из косвенных методов определения величины риска может быть ***оценка совокупных затрат***, необходимых для ликвидации негативных последствий в программах, системе или внешней среде, проявившихся в результате конкретного рискового события.

Процессы анализа и сокращения рисков должны сопутствовать основным этапам разработки и обеспечения качества компонентов и программных комплексов. Эти процессы могут быть отражены ***этапами работ и процедур***, которые рекомендуется выполнять при поддержке производства компонентов и комплексов программ, и могут служить основой для разработки соответствующих планов работ при управлении и сокращении рисков (рис. 1.7):

- анализ рисков следует начинать с подготовки детальных исходных требований и характеристик комплекса программ, системы и внешней среды, для которых должны отсутствовать риски функционирования и применения;

- для управления рисками и их сокращения в проектах комплексов программ рекомендуется выделять три основных класса рисков: функциональной пригодности, конструктивных характеристик качества и нарушения ограничений ресурсов при реализации процессов производства программ;
- в каждом классе целесообразно анализировать несколько категорий наиболее важных рисков, которые упорядочивать по степени опасности, угрозам для проекта, обусловленным ограничениями ресурсов, дефектами и/или недостаточным качеством производства программ;
- контрмеры для сокращения рисков рекомендуется анализировать и применять последовательно, начиная с ликвидации наиболее опасных исходных причин – угроз, затем проводить анализ и уменьшение уязвимости компонентов и комплекса в целом, а при недостаточности этих контрмер воздействовать непосредственно на уменьшение итогового ущерба – риска в жизненном цикле комплекса программ и системы;
- процессы устранения рисков должны завершаться процедурами мониторинга, сопровождения и конфигурационного управления изменениями версий компонентов и комплексов программ высокого качества с минимальными допустимыми рисками.

Управления рисками предполагает ясное понимание внутренних и внешних причин и реальных источников угроз, влияющих на качество компонентов и комплекса программ, которые могут привести к большому ущербу. В результате анализа следует создавать план ***отслеживания изменения рисков в жизненном цикле комплекса программ***, который должен регулярно рассматриваться и корректироваться. Главной целью управления рисками является обнаружение, идентификация и контроль за редко встречающимися ситуациями и факторами, которые приводят к негативным – рисковым результатам. Это должно отражаться на применении регламентированных процессов, в которых факторы и угрозы рисков систематически идентифицируются, оцениваются и корректируются.

Одной из самых распространенных причин и опасных источников рисков, являются ***ошибки при оценке масштаба – размера*** программного комплекса.

Требования к допустимым рискам и документам требований к комплексам программ должны включать:

- риски при формировании требований к характеристикам компонентов и сложных программных комплексов:
  - риски оценки масштабов комплексов программ;
  - формирование допустимых рисков требований к характеристикам компонентов и программных комплексов;
  - выделение, идентификацию, анализ угроз и рисков программного комплекса;
  - оценивание опасности угроз и рисков и выбор контрмер для их сокращения;
  - сокращение или ликвидацию опасных рисков комплекса программ;
  - контроль, регистрацию, мониторинг и утверждение допустимого интегрального риска комплекса программ;
- требования к допустимым рискам применения сложных программных комплексов;
- стандарты требований по управлению рисками в жизненном цикле программных комплексов;
- соглашение между заказчиком и разработчиками о документировании требований, содержания и применения программного комплекса;
- техническое задание на проект и исходные требования к программному комплексу:
  - организация документирования требований к компонентам и программным комплексам;
  - детальные требования к функциям и характеристикам конкретного программного комплекса.
  - документирование требований к программным компонентам и комплексам:
  - документирование требований технического задания на проект программного комплекса;
- примеры документирования требований к функциям и характеристикам программного комплекса:
  - документ общих системных требований к программному комплексу;
  - документ детальных требований к конкретному программному комплексу.

Рис 1.7

Эти ошибки, чаще всего, бывают случайными – непредумышленными, вследствие недостаточной компетентности заказчика или разработчика-поставщика. Однако в некоторых случаях в превышении значения согласованного масштаба заказанного продукта могут быть заинтересованы разработчики для получения больших ресурсов от заказчика, а уменьшенную оценку масштаба могут стремиться представить заказчики для сокращения выделяемых затрат на разработку программного продукта. Величина оцененного и согласованного между заказчиком и разработчиком масштаба комплекса программ непосредственно отражается на:

- ***бюджете и трудоемкости*** разработки и обеспечения всего жизненного цикла программного комплекса;
- ***затратах времени, сроках*** создания и всего жизненного цикла реализации и применения комплекса программ;
- потребности в ***численности и квалификации*** специалистов для реализации проекта и создания программного продукта в соответствии с требованиями заказчика.

Эти три ключевых характеристик обычно тесно связаны и могут изменяться в процессах разработки проекта заказчиком или разработчиком в сторону увеличения или уменьшения в соответствии с их интересами, целями и возможностями. Для снижения возможного ущерба – рисков применяются ***оценки, контроль и мониторинг рисков, а также различные контрмеры***. Уменьшение рисков должно производиться путем максимально возможного приближения проекта к требованиям заказчика и к выделенным ресурсам или путем снижения этих требований и увеличения заказчиком ресурсов на комплекс программ. В крупных проектах систем, использующих комплексы программ, риски могут быть обусловлены дефектами функциональных характеристик программ, а также недостатками систем и внешней среды, в которой они используются.

Риски могут проявляться в процессах проектирования и производства при изменении и развитии комплексов программ и при применении готового программного продукта по прямому назначению. Это приводит к необходимости анализа рисков функционирования в условиях, различающихся:

- источниками и причинами ***угроз*** и опасного проявления рисков;

- классами, категориями, ***уязвимостью*** комплекса программ и системы, вероятностью проявления и величиной последствий рисков;
- возможными и реализуемыми ***контрмерами*** для предотвращения и сокращения рисков и их эффективностью.

Неопределенность оценок масштаба комплексов программ является одним из важнейших факторов, ***влияющим на риски всего жизненного цикла проекта***. Точность оценки размера нового комплекса программ значительно повышается после формулирования начальных требований и спецификаций заказчиков, проведения анализа требований после завершения разработки предварительного проекта. На этапе концепции проекта ошибки оценки размера уменьшаются приблизительно до 40%. Это вполне объяснимо, поскольку еще не уточнены структура и многие детали проекта. Эти вопросы могут быть разрешены во время разработки структуры и спецификаций требований к комплексу программ, и тогда можно оценить его размер с точностью до 15 – 20% .

Как только структура комплекса будет разбита с учетом выделения самых нижних, доступных ***уровней компонентов*** может быть создан более точный показатель размера комплекса программ. После завершения разработки и подтверждения проектных спецификаций при детальном проектировании комплекса программ может быть определена структура внутренних данных и функции программных компонентов. На этом этапе ***оценка размера комплекса*** может составить около 10%. Уточнения размеров комплекса и компонентов, могут быть решены к концу детального проектирования, однако при этом сохраняется неопределенность оценки размера комплекса около 5 – 10%, связанная с тем, насколько хорошо программисты понимают спецификации, в соответствии с которыми они должны кодировать программы.

Если при оценке масштаба проекта его величина в договоре определена ***недостаточной – заниженной***, то основной ущерб – риски достаются разработчикам. Они вынуждены принимать жесткие меры для выполнения плановых сроков, выделенного бюджета работ при относительно малом числе специалистов, что угрожает рисками срыва сроков и нарушения стоимости проекта. Недооценка размера комплекса программ и ресурсов для его реализации может резко ***увеличивать число дефектов*** в программах. Кроме того, ***ограниченные ресурсы*** для реализации в полном объеме функциональной пригод-

ности, могут отражаться на ее качестве и удобстве применения, а также на конструктивных характеристиках: на безопасности, надежности, качестве взаимодействия с внешней средой и с пользователями, качестве документации и других эксплуатационных факторах.

Для продолжения разработки проекта после оценки рисков масштаба комплекса программ необходимо выполнить цикл поэтапного определения и формирования совокупности спецификаций требований к проекту комплекса программ. Первым этапом является создание концепции проекта и ***комплекса первичных требований*** к иерархическому набору функций, на которые могут быть разбиты предполагаемые фактические компоненты комплекса. В дальнейшем ***декомпозиция*** может детализироваться, формируя упрощенный или более точный уровень абстракции и взаимодействия компонентов. ***Для устранения или снижения рисков*** до допустимых пределов может быть необходимо изменение требований к функциональной пригодности, к конструктивным характеристикам и доступным ресурсам.

Обычно заказчики и разработчики первоначально устанавливают требования к каждой характеристике компонента или комплекса программ без учета относительных затрат на их достижение, а также без детального анализа их совместного влияния на полную функциональную пригодность и риски у потребителей. Это может приводить к значительным перекосам и ***несбалансированным значениям требований к отдельным, взаимосвязанным характеристикам***, на которые не рационально используются ограниченные ресурсы, или к не адекватно низким их значениям. В проектах это может угрожать значительным повышением стоимости и рисков и/или снижением конкурентоспособности создаваемого программного продукта из-за недостаточного уровня отдельных показателей качества.

Атрибуты качества компонентов и комплексов программ имеют различные меры и шкалы, вследствие чего они в большинстве своем непосредственно ***не сопоставимы между собой***. Они предварительно выбираются и согласовываются с заказчиком при последовательном, почти независимом анализе каждого атрибута качества, для использования в контракте и технических заданиях. Для обобщенного оценивания качества необходим учет относительного влияния каждой конструктивной характеристики ***на функциональную пригодность***. При этом не всегда учитываются ресурсы для их реализации в кон-

кретном комплексе. Это часто приводит к выдвижению ряда не рациональных требований, которые значительно отличаются либо по степени влияния на функциональную пригодность, либо по величине ресурсов, необходимых для их реализации. Для целенаправленного эффективного управления рисками при проектировании целесообразно иметь механизм объединения разнородных характеристик в некоторый интегральный показатель, отражающий их совокупное влияние на функциональную пригодность. Таким образом, при разработке требований к характеристикам проекта выявилась проблема анализа системной эффективности компонентов и комплекса и обобщения его характеристик, а также оценивания совместного влияния конструктивных характеристик на функциональную пригодность с учетом затрат на их реализацию.

Эти данные могут использоваться, прежде всего, ***как ориентиры*** для селекции и исключения из требований конструктивных характеристик качества с особенно низкими обобщенными приоритетами рисков, в наименьшей степени влияющих на функциональную пригодность и не оправдывающих больших затрат на реализацию. Анализ оставшихся характеристик качества и рисков может проводиться для выделения завышенных требований, а также, возможно, для снижения их значений и приближения влияния к средним значениям. Кроме того, сравнительный анализ обобщенных приоритетов на основе отношения влияния на качество и затраты позволяет выделять атрибуты качества, отличающиеся большими затратами – рисками, не оправданными их степенью воздействия на функциональную пригодность. Подобные процедуры могут завершать разработку требований к свойствам, характеристикам качества и допустимым рискам при детальном проектировании сложных комплексов программ.

Выше анализировалось преимущественно ***изменение функциональной пригодности и снижение риска*** при совершенствовании конструктивных характеристик программ. Однако для заказчика и пользователей доминирующее значение могут иметь номенклатура и особенности реализации некоторых основных функций комплекса программ, которые, как правило, требуют наибольших затрат и определяют основной эффект или возможные риски от применения комплекса программ, а также потенциальный уровень спроса на рынке. На основе анализа и оценивания характеристик масштаба, набора

требований, возможных затрат ресурсов и ***значений рисков следует определять***:

- целесообразно ли продолжать работы над конкретным проектом или следует его прекратить вследствие больших рисков, недостаточных ресурсов специалистов, времени или трудоемкости (бюджета) разработки;
- при наличии достаточных ресурсов следует ли провести маркетинговые исследования для определения рентабельности полного выполнения проекта и создания программного продукта для поставки на рынок;
- достаточно ли полно и корректно формализованы концепция и требования к проекту, на основе которых проводились оценки затрат и рисков, или их следует откорректировать и выполнить повторный анализ с уточненными исходными данными.

Требования к допустимым рискам применения сложных программных комплексов

Требования к сокращению рисков, связанных с результатами разработки и применения программного комплекса, должны реализоваться формализованным процессом, позволяющим ***выделять, систематически идентифицировать, оценивать и смягчать факторы, угрозы и величину последствий возможных рисков***. Для этого при управлении проектом следует: идентифицировать угрозы и риски, имеющие как внешние, так и внутренние причины; проводить их количественную оценку; разработку откликов и контрмер для сокращения рисков и контроль реализации откликов (см. рис. 1.7).

Обеспечение минимальных допустимых рисков требований функциональной пригодности имеет доминирующее значение и изменения других классов рисков обычно должны быть, в первую очередь, подчинены сокращению рисков функционирования системы и комплекса программ. Поэтому анализ рисков и возможных угроз целесообразно проводить систематизированно, начиная с установления требований к допустимым рискам функциональной пригодности. Ущерб вследствие ошибок функциональных требований к проекту программного комплекса может проявляться двумя видами рисков: недостатками достигнутых характеристик комплекса и рисков от нарушения ограниченности доступных и используемых ресурсов в его жизненном цикле.

В жизненном цикле важнейшим риском является ущерб при невыполнении комплексом программ, назначения и функций главной характеристики качества – ***функциональной пригодности,*** формализованной в требованиях заказчика. В зависимости от назначения, функций, критичности требований к системе и программному комплексу может требоваться либо полная ликвидация определенных или всех видов рисков, либо сокращение некоторых из них до допустимых пределов, либо игнорирование некоторых и сохранение на достигнутом уровне ущерба вследствие нарушения требований заказчика к программному комплексу.

Для выполнения требуемых функций комплекса программ необходима ***адекватная исходная информация от объектов внешней среды***, содержание которой должно полностью обеспечивать реализацию декларированных функций. Полнота формализации номенклатуры, структуры и качества входной информации для выполнения требуемых функций является одной из важных составляющих при определении функциональной пригодности и сокращении рисков применения комплекса программ в соответствующей внешней среде. Степень покрытия всей выходной информацией: целей, назначения и функций программного комплекса для пользователей, следует рассматривать как ***основное требование к допустимым рискам функциональной пригодности***. Прослеживание и оценивание адекватности и полноты состава выходной информации снизу вверх к назначению комплекса программ должны завершать выбор базовых характеристик функциональной пригодности, независимо от сферы применения системы.

При начальном формировании требований к функциональной пригодности комплекса программ практически невозможно достоверно ***предусмотреть сбалансированное выделение каждого вида ресурса*** для полной реализации каждой требуемой характеристики сложного программного комплекса. Кроме того, требования заказчика к функциям всегда субъективны и не стабильны, что также отражается на изменении рисков при разработке и модификациях комплексов программ. При этом некоторые характеристики в реальном проекте могут приобретать значения более высокие, чем действительно требуются, на что нерационально расходуются ресурсы, а другие – не удовлетворять требованиям контракта и технического задания. Для разрешения этого противоречия основное значение имеет

деятельность менеджера рисков, который должен быть способен прогнозировать, проводить поэтапный анализ, контроль, оценивание и мониторинг возможных и реальных отклонений от требуемых характеристик программного комплекса и используемых ресурсов, ***управление контрмерами и последовательное изменение их для сокращения интегрального риска*** всей системы.

Требования к оценке и сокращению рисков вследствие недостаточных характеристик качества комплексов программ должны сопровождать весь их жизненный цикл. Для этого необходимо контролировать области возможного возникновения рисков, оценивать вероятности их проявления, виды и степень влияния угроз, которые следует минимизировать как можно раньше по мере их возникновения и обнаружения в процессах жизненного цикла комплекса программ. Основной эффект по снижению рисков вследствие недостаточных характеристик должен достигаться на начальных этапах разработки, когда возможно предотвращение или сокращение многих из них с минимальными затратами времени и других ресурсов. Для этого в ***технологическом процессе производства*** необходимо использовать ***методы, которые включают***:

- определение ***ценности*** (приоритета) и выделение каждой ***требуемой характеристики*** для реализации необходимой функциональной пригодности программного комплекса и системы;
- определение приоритетов характеристик качества, компонентов и этапов производства, которые имеют потенциальные технические, стоимостные или плановые риски;
- оценивание вероятности каждого ***вида угроз*** характеристик качества, потенциальной величины и вероятности их возможного негативного воздействия на каждую характеристику функциональной пригодности системы и программного комплекса;
- оценивание ***уязвимости*** и последствий дефектов каждой характеристики и затрат ресурсов для восстановления требуемой функциональной пригодности системы и программного комплекса при проявлении рисков;
- систематизацию, документирование и оценивание эффективности доступных методов, средств и ресурсов ***контрмер*** для сокращения рисков функциональной пригодности и выделенных характеристик комплекса программ;

- планирование и разработку решений ***по контрмерам*** для обеспечения допустимого уровня интегрального риска функциональной пригодности и характеристик системы, в том числе, возможно, за счет ***изменения требований*** к программному комплексу, системе и/или доступных ресурсов;
- оценку ***вероятности сокращения рисков*** характеристик до допустимых пределов, при реализации процессов разработки и всего жизненного цикла программного комплекса с учетом доступных ресурсов.

Улучшение каждой характеристики требует затрат ресурсов, которые в той или иной степени должны отражаться на основной характеристике комплекса программ – ***на функциональной пригодности***. Эти характеристики имеют значение для проекта постольку, поскольку они обеспечивают требуемое качество реализации основного назначения и функций программного комплекса. При выборе конкретных допустимых характеристик следует учитывать возможные затраты ресурсов на их достижение и на результирующее повышение функциональной пригодности, желательно, в сопоставимых экономических единицах, в тех же мерах и масштабах. Такое, даже приблизительное, качественное сравнение эффекта и затрат позволяет избегать многих ***не рентабельных завышений требований*** к отдельным характеристикам, которые не достаточно отражаются на адекватном улучшении функций программного комплекса. Поэтому для каждого проекта необходимо ранжировать характеристики (приоритеты) и выделять, прежде всего, те, которые могут в наибольшей степени улучшить функциональную пригодность и снижать риски для конкретных целей системы.

Решение этих задач должно быть направлено на обеспечение высокой функциональной пригодности программного комплекса ***путем сбалансированного улучшения остальных характеристик в условиях ограниченных ресурсов***. Для этого в процессе системного анализа при подготовке технического задания и требований спецификаций, значения и риски характеристик, должны выбираться с учетом опасности их влияния на функциональную пригодность. Излишне высокие требования к отдельным характеристикам, требующие для реализации больших дополнительных трудовых и вычислительных ресурсов, целесообразно снижать, если они слабо влияют на основные, функциональные характеристики программного продукта.

Для управления требованиями и рисками с целью минимизации и выделения наиболее опасных из них необходимо сопоставлять вероятности (частости) и последствия проявления как рисков функциональных характеристик комплекса программ, так и рисков доступных ресурсов. Для каждого проекта эти виды рисков могут различно влиять на ***интегральный ущерб*** – риск комплекса программ, отражающийся на общем риске системы. Применяемые методы оценивания и анализа величин и вероятности рисков должны позволять определять приоритеты видов рисков с целью выделения соответствующей доли ***ресурсов на контрмеры*** для сбалансированного сокращения негативных последствий различных видов рисков.

Прямые риски, обусловленные ошибками и дефектами заданных требований экономических характеристик, могут вызвать ущерб заказчику при ***завышении*** стоимости проекта относительно реально необходимой или ущерб разработчикам, если стоимость ***оценена недостаточной*** для его успешной реализации. Эти риски могут уменьшаться при последовательном уточнении размера комплекса программ на этапах формирования требований, предварительного и детального проектирования, однако они не полностью учитывают реальное влияние ограничений ресурсов на процессы и риски производства конечного программного продукта. Поэтому ***риски нарушения ограничений доступных ресурсов проекта*** целесообразно оценивать по их последующему отражению на степень возможной реализации требований заказчика в конечном программном продукте. По мере уточнения размера – масштаба и развития проекта комплекса программ эти риски обычно уменьшаются, однако их влияние может оставаться существенным в процессе всего жизненного цикла.

Требования по управлению рисками в жизненном цикле программных комплексов регламентированы международным стандартом **ISO 16085.** Он включает особенности обеспечения, мониторинга качества и функциональной безопасности сложных комплексов программ на базе анализа и сокращения рисков процессов, регламентированных стандартом **ISO 12207**. В стандарте **ISO 15504** содержится раздел, определяющий регламентирование и планирование процессов выявления и устранения совокупности различных рисков на протяжении всего жизненного цикла. Поэтапное, иерархическое снижение интегрального риска проекта программного комплекса при использовании выбранной стратегии может требовать ее корректировки в за-

висимости от достигаемого эффекта и требуемых затрат на сокращение определенных рисков, необходимых для повышения качества, надежности и функциональной безопасности. Для решения этих задач ***стандартами рекомендуются последовательные процедуры***:

- выявление и идентификация относящихся к комплексу программ рисков как в исходном состоянии и требованиях к проекту, так и на последовательных этапах его выполнения: по характеристикам качества, графикам производства, трудоемкости, ресурсам и техническим рискам;
- анализ вероятности проявления, причин, взаимозависимости видов и последствий рисков, чтобы сбалансировать и распределить их приоритеты, на основании которых будут отводиться ресурсы на различные контрмеры для снижения этих рисков;
- определение целесообразной стратегии и области управления каждым видом риска или их наборами в проекте в соответствии с политикой на предприятии, а также со степенью влияния, вероятностью и типами рисков, которые должны выявляться и которыми следует управлять;
- для каждого вида риска (или набора рисков) определение метрики, отражающей изменения в состоянии рисков в зависимости от деятельности по их сокращению, которые должны характеризовать изменения в вероятности проявления, последствиях и временных диапазонах проявления рисков;
- реализация разработанной стратегии управления рисками, аттестация результатов и оценка успешности стратегии в контрольных точках жизненного цикла проекта.

Документирование требований к программным компонентам и комплексам

Общие требования к документации сложных комплексов программ и компонентов определены в стандарте **ISO 9126** в разделе практичность. ***Требования к практичности*** комплекса, функциям, характеристикам и документам, включают их понятность и простоту использования, зависят от его назначения и функций. Они могут формализоваться заказчиками ***набором свойств***, необходимых для обеспечения удобной и комфортной эксплуатации программного комплекса достаточно квалифицированными пользователями. Для обеспечения полноценного изучения процессов применения комплекса специалистами

необходима ***эксплуатационная и технологическая документация***, требования к которой должны устанавливаться заказчиком и входить в ***состав основных требований к программному комплексу***. Ее объем существенно зависит от назначения и функций продукта и может быть задан на основе анализа прецедентов подобных успешных проектов. Для некоторых программных комплексов, подлежащих широкому тиражированию, могут быть желательны адекватные по содержанию электронные учебники, требования к объему и функциям которых также целесообразно оценивать по прецедентам. Следует учитывать, что малый объем эксплуатационной документации может снизить качество и полноту использования функций сложного комплекса программ, а очень большой объем – также может ухудшить эксплуатацию из-за трудности выделения из множества второстепенных деталей и освоения наиболее существенных свойств и особенностей применения комплекса.

Общее руководство по документированию проекта и требований к программному комплексу должно быть в составе ***соглашения между заказчиком и разработчиками*** (см. рис. 1.7). В базовом документе о функциях, образе и границах документов комплекса программ должны содержаться требования заказчика, а дополнительные пожелания пользователей могут иногда представляться в виде вариантов использования комплекса. Подробные функциональные и нефункциональные требования должны быть зафиксированы в ***спецификации требований*** к программному комплексу. Однако если не собрать все требования вместе в виде структурированного и конкретного документа и не ознакомить с ним всех заинтересованных в проекте лиц, то у них не будет уверенности, что они согласны со всеми положениями спецификации требований и смогут их выполнить.

Документация исходных требований значительно ***влияет на достигаемое качество*** сложных комплексов программ, трудоемкость и длительность их создания. Организация документирования должна определять стратегию, стандарты, процедуры, распределение ресурсов, планы создания, изменения и применения документов требований на программы и данные систем. Для этого в общем случае должны быть выделены ***руководители и коллективы специалистов, которые должны планировать, утверждать, выпускать, распространять и сопровождать комплекты достоверных документов на программный комплекс***. Они должны стимулировать разработчиков

компонентов, программных комплексов и их корректировок, осуществлять непрерывное, регламентированное документирование процессов и результатов своей деятельности, а также контролировать полноту и качество документов. Структура документации и формы отдельных документов, используемых для совершенствования требований к программам, ***должны позволять***:

- точно документально описывать и идентифицировать требования к каждой оформленной версии программных модулей, компонентов и комплексов в целом в любое время на всем протяжении их жизненного цикла;
- надежно учитывать и регистрировать все анализируемые, подготавливаемые и проведенные корректировки требований в версиях модулей, компонентов и комплекса;
- снабжать руководителей проекта обобщенной и детальной информацией для принятия решений на изменения требований к программам, а также для контроля выполнения принятых решений;
- обеспечивать заказчиков и пользователей объективными сведениями о наиболее существенных корректировках требований и о новых версиях компонентов и программного комплекса.

В документах целесообразно использовать ***иерархическую систему идентификации требований*** к версиям программного комплекса, его модулям, компонентам и описаниям данных. Присвоение идентификаторов для составляющих проекта, вплоть до утвержденных требований к версиям программных модулей, компонентов и комплексов, целесообразно осуществлять централизованно группой ***конфигурационного управления***. Некоторый произвол в идентификации компонентов может допускаться только в процессе подготовки корректировок программ и данных до комплексирования их в версию программного комплекса, предъявляемую на испытания.

Техническое задание на проект и исходные требования к программному продукту целесообразно формировать на основе общих положений программной инженерии и должны содержать поэтапно уточняющиеся, детализирующиеся и дополняющиеся при детальном и рабочем проектировании разделы:

- титульный лист с утверждающими и согласующими подписями заказчика и разработчиков;
- полное наименование проекта программного комплекса;

- назначение, цель разработки или модернизации программного комплекса;
- основание для выполнения и финансирования проекта;
- организация – заказчик проекта;
- предприятие – головной исполнитель и соисполнители проекта;
- общие сроки выполнения всего проекта программного комплекса;
- общие технические требования, перечень стандартов и базовых нормативных документов для выполнения проекта;
- общие требования к программному комплексу; требования к функциям и основным характеристикам качества;
- требования к внешней среде применения комплекса;
- специальные требования к аппаратной и операционной платформам для реализации программного комплекса;
- требования к структуре, оформлению и содержанию эксплуатационной и технологической документации;
- этапы и график выполнения основных работ проекта;
- ожидаемые результаты проекта и форма их представления;
- порядок контроля исполнения проекта и приемки результатов работы.

При разработке требований к программным комплексам, кроме основных целей, назначения и функций, важно учитывать и формулировать содержание полного множества характеристик, каждая из которых может влиять на успех применения программного комплекса. Для уменьшения вероятности случайного пропуска важного требования заказчикам и пользователям целесообразно иметь типовые ***шаблоны документов***, содержащих структурированные наборы требований, которые можно целеустремленно сокращать и/или адаптировать, обеспечивая ***целостность всей совокупности требований*** для конкретных комплексов программ.

Выходные проектные данные для применения программного комплекса должны быть ***документально оформлены*** и выражены в требованиях заказчика так, чтобы их можно было проверить и подтвердить соответствие входным проектным требованиям. Эти данные должны содержать ***критерии приемки программного продукта*** заказчиком или ссылки на них, а также идентифицировать те характеристики, которые являются критическими для его надежного и безо-

пасного функционирования и применения. При работе над любым проектом необходимо достигать согласованности решений по каждому набору и значениям требований до начала их реализации разработчиками.

Для разработчиков особенно важно ***формализовать требования в документах и согласовать их с заказчиком при утверждении контракта и технического задания на проект***. Требования к характеристикам, утвержденные после предварительного проектирования, могут быть закреплены в техническом задании ***как обязательные для детального и/или рабочего проектирования***. Эти данные должны использоваться при последующем оценивании функций и характеристик качества и их сопоставлении с требованиями в процессах тестирования, квалификационных испытаний или сертификации программного комплекса. Однако для сложных и дорогих проектов может потребоваться ***уточнение документов требований*** при детальном проектировании с позиции улучшения соотношения значений качества и затрат ресурсов, которые необходимы или допустимы для их реализации.

На основе спецификации требований должны составляться и документироваться планы разработки проекта, а также ***тестирования системы, комплекса программ и эксплуатационной документации***. Она должна содержать описание и сценарии поведения системы при различных условиях. Детали дизайна, сборки, тестирования или управления проектом, документированные в спецификации, не должны противоречить ограничениям разработки. Будучи конечным хранилищем требований к программному продукту, спецификация требований должна быть ясной и понятной, чтобы у разработчиков и заказчиков не оставалось ни малейших возможностей для разночтения. Состав стандартизированных характеристик программных комплексов должны быть отдельным, обязательным разделом в общей спецификации требований, итерационно формируемыми на этапах жизненного цикла и контролируемыми при испытаниях программного комплекса.

Чтобы было легко отслеживать и модифицировать содержание документов, каждое функциональное требование должно быть представлено уникально и неизменно. Это позволит ссылаться на определенные требования в запросе на изменения, в хронологии изменений, в перекрестных ссылках или матрице для прослеживания реализации

требований. При этом также упрощается многократное использование требований к модулям и компонентам в нескольких проектах. Для эффективного управления и модификации требований с учетом сценариев и процедур их тестирования, должна использоваться технология и инструментарий ***управления конфигурацией комплексов программ***.

Документирование требований к функциям и характеристикам комплексов программ

В составе требований, ориентированных на сложные проекты, особенно важно полно и детально регламентировать необходимые характеристики комплексов программ. Использование при адаптации в качестве исходного, максимального набора процедур разработки и состава требований, позволяет предотвращать пропуск некоторых из них, существенных для развития конкретного проекта. Результаты комплексной разработки требований следует ***документировать и учитывать***, при подготовке и утверждении заказчиком технических заданий и спецификаций требований на программный комплекс. Такая технология систематизированной, итерационной разработки требований к комплексу программ позволяет избежать случайных пропусков, пробелов и неопределенностей в составе и содержании утверждаемых заказчиком документов в технических заданиях и спецификациях.

Стандартизированные в **ISO 9126** характеристики качества комплексов программ различаются по степени влияния на системную эффективность их применения по прямому назначению. Высшие приоритеты, естественно, должны присваиваться свойствам и атрибутам функциональной пригодности для достижения ***стратегических целей*** использования программного комплекса. Все остальные стандартизированные характеристики, в той или иной степени, должны способствовать обеспечению ***тактических целей*** выбранных конструктивных требований. Разработка требований к свойствам и характеристик качества, естественно, должна начинаться с определения и документирования группы показателей, непосредственно отражающих ***функциональную пригодность***. Компоненты требований функциональной пригодности очень разнообразны и их целесообразно распределять ***на две части***: унифицируемую по структуре и содержанию для различных по назначению комплексов программ, и уникальную –

непосредственно связанную со специфическими функциями и целями применения программного комплекса.

Для ***функциональных возможностей*** (стандарт **ISO 9126**) следует регистрировать приоритеты реализации соответствующих требований, а также указывать документы, где детализированы особенности этих свойств для конкретного проекта. Требования к характеристике ***корректность*** могут представляться в виде описания двух основных свойств, которым должны соответствовать все программные компоненты и комплекс в целом. Первое требование состоит в выполнении определенной степени прослеживаемости и верификации сверху вниз реализации требований технического задания при последовательной детализации описаний программных компонентов вплоть до текстов и объектного кода модулей программ. Второе требование заключается в выборе степени и стратегии покрытия тестами требований структуры и функций программных модулей и компонентов, достаточных для функционирования программного комплекса с необходимым качеством и точностью результатов при реальных ограничениях ресурсов на тестирование.

Для освоения и применения широкого состава функций, свойств и атрибутов характеристик при разработке требований и документов к программному комплексу необходимы квалифицированные специалисты и значительный объем работ. Полностью это может быть рентабельно при создании достаточно крупных проектов. В этих случаях ***обсуждение и согласование между заказчиком и разработчиком*** всего состава ***документов требований*** к функциям, свойствам и рациональным значениям ансамбля характеристик качества, позволяет избегать как нецелесообразное завышение требований и использования ресурсов, так и не оправданное снижение требований к отдельным характеристикам. Это может негативно отражаться на функциональной пригодности программного комплекса и технико-экономических характеристиках всего проекта. При этом важно учитывать ограниченность ресурсов при выборе свойств и значений характеристик и ***необходимость компромиссов*** между ними вследствие многочисленных связей и взаимовлияний.

Документацию, формализующую требования к сложному программному комплексу, целесообразно делить на ***две части***. Первый документ – ***системные требования***, предназначен для определения роли, назначения и функций программного комплекса в соста-

ве системы для определенной внешней среды. Он используется, прежде всего, заказчиком и менеджером – главным конструктором при организации жизненного цикла сложного комплекса программ. Этот документ подлежит тщательному согласованию между заказчиком и разработчиками и ***утверждению*** заказчиком при любых изменениях требований. Второй документ определяет более ***детальные требования*** к реализации комплекса программ разработчиками. Формализация и реализация этих требований иногда частично может проходить без обязательного согласования с заказчиком, но документ должен служить основой для деятельности всего коллектива разработчиков. Требования в этих двух документах могут несколько повторяться, но это может быть удобно, так как их основные пользователи существенно различаются квалификацией и ответственностью за успех проекта. Ниже приводятся ***примеры шаблонов двух документов***, ориентированных на разработку сложных программных продуктов высокого качества реального времени.

Документ общих системных требований к программному комплексу должен содержать следующие сведения:

- требования к назначению и архитектуре системы, содержащие идентификацию и функции основных компонентов системы, их назначение, статус разработки, аппаратные и программные ресурсы;
- требования проекта системы к функциям программного комплекса, как к целому в общей архитектуре системы;
- цели, назначение программного комплекса, потребности заказчика и потенциальных пользователей к его функциям и характеристикам в заданной системной среде применения;
- распределение системных требований по основным функциональным компонентам программного комплекса с учетом требований, которые обеспечивают заданные характеристики системы;
- требования к совместному целостному функционированию компонентов, описание и характеристики их динамических связей при применении комплекса программ;
- требования к унификации интерфейсов и базы данных компонентов комплекса программ;
- требования трассируемости функций компонентов программного комплекса к требованиям проекта системы;
- спецификацию требований к функциям, качеству, надежности и безопасности программного комплекса;

- производные требования к компонентам программного комплекса и требования к интерфейсам между системными компонентами, элементами конфигурации комплекса программ и аппаратуры;
- требования к применению базовых стандартов и к формированию профиля стандартов для проекта программного комплекса.

Документ детальных требований к конкретному программному комплексу для обеспечения его разработки, сопровождения и модификации должен содержать:

- общие требования к программному комплексу:
  - перечень объектов внешней среды применения программного комплекса (технологических объектов управления, подразделений предприятия и т.п.), при управлении которыми должен решаться требуемый комплекс функциональных задач;
  - требования к операционной среде функционирования программного комплекса;
  - требования к составу и качеству реализации функций и характеристик программного комплекса;
  - распределение функций между персоналом, программными и техническими средствами при различных ситуациях решения требуемого комплекса функциональных задач;
  - требования к периодичности и допустимой продолжительности решения функциональных фрагментов комплекса задач в реальном времени;
  - требования к надежности и безопасности функционирования системы и программного комплекса;
  - требования к производительности системы и программного комплекса с заданным качеством;
  - требования к возможности адаптации функций комплекса программ к различной внешней среде и условиям применения;
  - требования к оформлению результатов и документов программирования и текстов программных модулей и компонентов;
- требования к входной информации комплекса программ должны содержать:
  - источники информации и их идентификаторы;
  - перечень и описание входных сообщений (идентификаторы, формы представления, регламент, сроки и частота поступления данных);
  - перечень и описание структурных единиц информации вход-

ных сообщений или ссылки на документы, содержащие эти данные;

- требования к выходной информации комплекса программ должны содержать:
  - назначение и потребители выходной информации;
  - перечень и описание содержания выходных сообщений;
  - регламент и периодичность выдачи результирующей информации;
  - допустимое время задержки отклика на результаты решения определенных задач;
  - степень покрытия тестами компонентов и комплекса программ и допустимый уровень проявления дефектов и ошибок при их функционировании;
- требования к необходимым ресурсам на разработку, ввод в действие и обеспечение функционирования при применении программного комплекса;
- требования, гарантирующие качество применения программного комплекса, обеспечение надежности и безопасности системы и внешней среды;
- обоснование выбора оптимального варианта требований к содержанию и приоритетам реализации совокупности функций программного комплекса;
- требования к структуре, составу компонентов и интерфейсам с внешней средой и пользователями;
- требования к эффективности реализации выбранного варианта требований к программному комплексу по использованию доступных вычислительных ресурсов;
- требования к плану и срокам реализации требований к программному комплексу;
- требования к составу и содержанию эксплуатационной и технологической документации на программный комплекс, достаточной для его применения, сопровождения и модификации;
- требования к методам и программе испытаний, приемки системы и программного комплекса заказчиком;
- требования к средствам обучения и уровню квалификации специалистов – пользователей программного продукта.

Лекция 1.6

ЭТАЛОНЫ ТИПОВ ТЕСТОВ И ИЗМЕНЕНИЯ ТРЕБОВАНИЙ К КОМПЛЕКСАМ ПРОГРАММ

Формализация эталонов типов тестов программного комплекса и компонентов

Объектами тестирования далее рассматриваются сложные комплексы программ для систем обработки информации и управления, разрабатываемые организованными коллективами (командами) квалифицированных специалистами разных категорий. ***Эталоны для тестирования*** включают процессы и объекты, которые определяют свойства, характеристики и качество компонентов и комплексов программ**.** Реализация тестирования может производиться разными методами и ***независимыми специалистами*** – программистами, интеграторами и тестировщиками, что позволяет использовать результаты их деятельности для сравнения одних и тех же описаний программ, представленных на языках программирования и на языках ***описания эталонов – тестов***.

Первичное описание и реализация программных компонентов, а также ***мышление их разработчиков (программистов)*** осуществляется на основе требований, функций, характеристик структуры и исполнения программ. Оно существенно ***отличается от представлений и методов описаний тех же функций комплекса программ тестировщиками*** – создателями сценариев, процессов и типов ***эталонов*** тестирования. Они акцентируют свою деятельность на конкретных процедурах проверки функционирования, возможных результатах и взаимодействии компонентов комплекса программ. Это позволяет выявлять дефекты и повышать качество путем сопоставления двух методов и результатов описания ***одних и тех же*** функций и характеристик программ за счет того, что мала вероятность одинаковых ошибок в сценариях и реализациях тестов и в описаниях требований к функциям и характеристикам программ.

Совокупности требований и описаний типов тестов могут применяться ***как эталоны и вторая адекватная форма описаний функций и характеристик комплексов и компонентов программ***. Для обеспечения высокого качества программных комплексов и компонентов стратегию создания тестов различных типов целесообразно строить***, основываясь на требованиях к функциям и характеристикам*** комплексов программ (см. рис. 1.2). При разработке сложных комплексов программ необходимо иметь пригодные для применения тестирования ***эталоны – требования и/или сценарии использования функций*** программных компонентов и комплекса. Такая вторая форма описаний содержания программ должна служить для сквозной верификации спецификаций требований к тестам сверху вниз, а также сами подвергаться верификации на корректность соответствия исходным требованиям к компонентам программ разного уровня.

Для обеспечения эффективности затрат ресурсов на тестирование компонентов и комплексов оно должно быть интегрировано как можно раньше с основными процессами проектирования, разработки и сопровождения в жизненном цикле программных комплексов. По существу, требования к программному компоненту и тесты для проверки их адекватности и полноты ***отражают один и тот же объект***, но в разной форме. Поэтому сложность представительного описания тестов ***соизмерима*** со сложностью описания полной совокупности требований к функциям и характеристикам соответствующего программного компонента. Вследствие этого трудоемкость и другие ресурсы для разработки адекватных тестов ***должны соответствовать ресурсам и трудоемкости*** при создании и реализации корректных требований, определяющих полностью программный компонент. Однако на практике обычно это не учитывается, и выделяемые на тестирование ресурсы оказываются значительно меньше и недостаточными для полноценного тестирования требований к компонентам программ, что определяет их не полное соответствие требованиям и недостаточное качество.

Для обеспечения высокого качества программного комплекса и компонента параллельно с ***верификацией требований*** и разработкой корректировок следует разрабатывать и верифицировать ***спецификации и сценарии тестов***, отражающие методы и конкретные процедуры проверки реализации этих требований. Тестовые спецификации

могут использоваться для проверки согласованности, внутренней непротиворечивости и полноты реализации требований без исполнения программ. Для каждого требования к функциям программного комплекса (компонента), его архитектуре должна быть разработана спецификация требований к тестам, обеспечивающая проверку корректности, адекватности и возможности в последующем реализовать тестирование на соответствие этому эталону. Такая взаимная проверка корректировок функций комплексов (компонентов), отраженных в требованиях и в спецификациях тестов, обеспечивает повышение их качества, сокращение дефектов, ошибок, неоднозначностей и противоречий – рис. 1.8.

Тестовый эталон как продукт помогает организовать и скоординировать усилия сотрудников, разрабатывающих и тестирующих программный комплекс. Однако многие тестовые эталоны не ограничиваются этой исключительно важной задачей. Формат, структура и уровень детализации подобных эталонов типов тестов определяются не только соображениями эффективности тестирования, но и нормами, определяемыми спецификацией. Если разрабатывается программный продукт для военных систем, то обязательно должны поставляться тестовые эталоны заказчику и соответствующие их спецификации. То же самое касается и программных комплексов для медицины. Каждый из эталонов служит для поиска ошибок. Однако важно понимать, что даже если время, затраченное на его разработку, могло бы быть с большей производительностью использовано для анализа и тестирования программы, все равно придется выдержать стандарты и написать тестовый план и эталоны, полностью соответствующие требованиям заказчика или регулирующей организации на комплекс программ.

Тестовый эталон как инструмент содержится в требованиях официальных стандартов для планов тестирования, таких как **ANSI/ IEEE 829**. Это целый комплекс требований к спецификации тестовых эталонов, спецификации для самих тестов, журналам тестирования, разнообразным идентификаторам, к спецификации для тестовой процедуры, спецификациям ввода/вывода, а также специальные процедурные требования, зависимости между тестами, календарный план, описание распределения работ между персоналом, критерии приостановки и возобновления тестирования.

Эталоны типов тестов и изменения требований к комплексам программ должны включать:

- формализацию эталонов типов тестов программного комплекса и компонентов:
  - программные «оракулы» – неформальные знания и представлений тестировщиков на основе опыта;
  - модели потоков управления и потоков данных в модулях, компонентах и комплексе программ;
  - реальные объекты внешней среды и системы, являющиеся источниками и потребителями информации;
  - сценарии и наборы тестов от имитаторов на основе аттестованных моделей внешней среды;
  - программные модели динамических объектов внешней среды, источники и потребители информации комплекса программ;
- формализацию документов как эталонов тестов комплексов программ:
  - эксплуатационная документация как эталоны тестов;
  - технологическая документация как эталоны тестов;
  - совершенствование, контроль адекватности и качества эталонных тестовых документов при управлении проектом;
  - контроль, обеспечение и гарантированное сохранение документации;
- управление изменениями требований к комплексам программ:
  - квалификацию видов и причин изменения требований к комплексам программ;
  - оценки стоимости изменения комплексов программ;
  - планирование и утверждение изменений требований к комплексам программ;
- организацию изменений и сопровождения требований к комплексам программ:
  - стратегию сопровождения изменений требований к комплексам программ;
  - подготовку и специфицирование изменений требований к комплексам программ;
  - определение ресурсов и состава специалистов для реализации изменений требований к комплексам программ.

Рис. 1.8

Спецификации тестов должны обеспечивать дополнительный контроль корректности спецификаций требований к функциям и верификацию взаимодействия компонентов на соответствующем уровне описания комплекса программ. Разработка спецификаций тестов (эталонов) на основе спецификаций требований, создает базу для обнаружения, какие требования не тестировались или принципиально не могут быть проверены тестированием. Таким образом, ***верификация спецификаций требований тестов*** к функциям и характеристикам программных компонентов могут использоваться с ***двумя целями***:

- для разработки и проверки программ и интерфейсов взаимодействия программных компонентов разных уровней в комплексе программ;
- для создания требований к скоординированному комплексу тестов для проверки совокупности компонентов, обеспечивающих взаимную проверку реализации спецификаций требований комплексом программ.

Тестирование всегда предполагает компромисс между ограниченными ресурсами и заданными сроками, с одной стороны, и практически неограниченными требованиями к качеству результатов тестирования и программных компонентов, с другой. ***Оценка стоимости и затрат***, а также другие измерения процессов, связанных с оценкой ресурсов, необходимых для тестирования, как и оценка ***экономической эффективности тестирования*** на разных этапах и уровнях, основывается на практиках менеджмента проектов и используется для оценки и улучшения процесса тестирования. Разные эталоны, концепции и модели тестирования требуют разных затрат по времени и необходимым ресурсам. Учет соответствия между стоимостью или затратами, необходимыми для того или иного метода и эффективности эталонов тестирования является обязательной частью современного управления проектами программных комплексов.

В зависимости от размера, сложности и назначения комплекса программ для тестирования могут применяться различные типы ***эталонов***, обеспечивающих обнаружение и устранение дефектов и ошибок (см. рис. 1.2):

- целостный ***комплекс требований*** к функциям, характеристикам, архитектуре и качеству тестов, адекватно отражающих и покрываю-

щих полный ***набор требований з***аказчика или программного менеджера проекта к программным компонентам и комплексу;
- исходные объекты и процессы, которым должны соответствовать типы тестов, используемых для выявления дефектов и ошибок:

- ***программные «оракулы»*** – неформальные знания и представлений тестировщиков на основе опыта создания предшествовавших аналогичных компонентов и комплексов программ в реализованных проектах или специальных имитационных программах и правила их формирования***;***
- модели ***потоков управления и потоков данных,*** отражающие процессы последовательной обработки информации в компонентах и комплексе программ;
- ***реальные объекты внешней сре***ды и системы, являющиеся источниками и потребителями информации при применении комплекса программ;
- ***сценарии, процедуры*** и данные от имитаторов динамических тестов на основе аттестованных моделей внешней среды;
- ***программные модели функционирования динамических объектов*** внешней среды, отражающие источники и потребителей информации комплекса программ;
- ***комплекты документации***, отражающей все свойства, функции, характеристики и качество компонентов и комплекса программ, охваченный глубоким, детальным ***архивированием*** и ***конфигурационным управлением*** исходных, промежуточных и отчетных документов:
- ***эксплуатационная документация*** для пользователей, обеспечивающая квалифицированное применение комплекса программ по назначению;
- ***технологическая документация,*** отражающая архитектуру, содержание компонентов и свойства комплекса, обеспечивающая возможность их модификации и развития.

Представление тестов программных компонентов и комплексов на базе достоверных и полных к ним требований, предполагает использование трудоемких процессов формирования таких исходных эталонов. Это может быть необходимо при разработке программных компонентов для сложных комплексов программ с требованиями обеспечения и гарантирования высокого качества, надежности и безопасности функционирования критических систем. В более про-

стых проектах широко применяется более экономичный, полуформальный метод формирования тестов и выполнения тестирования на основе применения понятия тестовый «оракул».

Тестовый «оракул» – тип эталонов тестов для предсказания и оценки корректности поведения модуля или программного компонента, предназначенный для тестирования в определенных условиях. Предполагается, что тестировщик благодаря личному опыту и квалификации знает, или имеет конкурирующую, действующую или предшествующую версию программы, которая может имитировать основные варианты корректного поведения новой программы, предназначенной для тестирования. Тестировщику доверяется принятие решения о том, является ли сообщение об ошибке при исполнении программы достоверным и ее необходимо корректировать. Большинство методов исследовательского тестирования программных компонентов опирается в определенной степени на подходы, в которых в качестве «оракула» выступает тестировщик, хотя опыт такого тестирования включает в себя также применение в качестве «оракулов» результатов функционирования предшествующих аналогичных или «пилотных» моделей.

Когда документируется система тестов, то обычно пытаются точно описывать предстоящее тестирование. Даже точный «оракул» не всегда может рассказать все, что желательно знать о поведении тестируемой программы. Это относится к тестовым сценариям, начиная с исследовательских предположений и кончая написанными с помощью детальных сценариев последовательностей действий в тестах (скриптов). Тестовые сценарии, зафиксированные в скриптах, определяют действия, данные и ожидаемые результаты с некоторой неопределенностью. Неопределенность «оракула» – это, безусловно, нежелательное свойство при определении соответствия требованиям или рисков качества. В отличие от них, однозначные эталоны (требования) тестирования поддерживают воспроизводимые тесты, что важно для регрессионного тестирования, они позволяют сделать интерпретацию результатов тестирования менее зависимыми от человека, выполняющего тесты. Их можно контролировать, а другие специалисты могут изучить средства тестирования и понять, что и как было подвергнуто тестированию.

Следующие факторы, которые способствуют повышению качества, увеличению объемов документации и сложности тестов, целесообразно учитывать при тестировании по методу «оракулов»:

- мозг человека способен обрабатывать одновременно только конечное число элементов информации, поэтому порядок, время и данные должны быть четко организованы при тестировании, требуют исчерпывающего документирования, чтобы убедиться, что нужные события в системе происходят в нужное время;
- тестовые сценарии и процедуры, которые находятся в голове только у одного человека, порождают организационный риск и риск управления для всей группы тестирования комплекса программ;
- потребность в регрессионном тестировании и последовательной воспроизводимости тестов;
- потребность в формализации и стандартизации процесса тестирования и документации на программные компоненты и комплексы;
- уровень риска – системы с повышенными требованиями к безопасности и с особым целевым назначением должны работать очень точно при каждом использовании программ;
- жесткие временные рамки для выполнения тестов компонентов в комплексах реального времени;
- необходимость руководить неопытными тестировщиками и делиться знаниями – недостаточно квалифицированные специалисты при участии в тестировании системы, могут внести свою негативную лепту в процесс тестирования сложных критических систем.

Тестовое покрытие – это отображение пространства варьирования тестов на некоторую совокупность эталонов к функциям и характеристикам программ, предназначенных для тестирования. Покрытие можно отражать ***с помощью матрицы***, где по одной оси представлены тестовые ***эталонные сценарии***, а по другой ***требуемые функции***, характеристики, безопасность или риски комплекса или компонентов программ. При этом можно отслеживать степень покрытия эталона тестами, поддерживаемыми конфигурацией и сценариями использования программного компонента или комплекса. Подобные модели могут фиксировать ***потоки управления и/или потоки данных компонента***, последовательность состояний, в которых может находиться программный компонент или комплекс и последовательность их обработки.

Можно проводить оценку степени покрытия требований относительно этих моделей, проверяя, что каждому узлу и каждой связи в графе соответствует тестовый сценарий. Оценка покрытия требует

учета, в какой степени тесты нацелены на пространство требуемых функций и характеристик программного компонента или комплекса. Точность определения степени покрытия ограничена, но выделение сильных и слабых связей между тестами и пространством требований к функциям и характеристикам компонента позволяет акцентировать анализ на безопасности, допустимых рисках или некоторых характеристиках, особенно важных для конкретного проекта.

Однако точные сценарии и типы тестов на базе эталонных требований имеют обратную сторону. Описание точного тестового сценария обычно очень многословное и сложное. Чем точнее и полнее документированы требования и сценарии тестирования, тем точнее должно быть предвидение о качестве исполнения программного компонента. Точность способствует повторному использованию тестов при регрессионном тестировании и предполагает также возможность сопровождения и модернизации тестов. Чем более точно подготовлены сценарии тестирования и эталоны, тем больше усилий требуется для их изменения, когда в тестируемую программу вносятся корректировки.

Данные натурных экспериментов с объектами внешней среды в качестве требований и тестов могут подготавливаться заранее вне сеансов динамического тестирования программного комплекса, например, при отладке аппаратной части системы обработки информации. Эти данные отражают характеристики и динамику функционирования объектов, которые трудно или опасно подключать для непосредственного взаимодействия с недостаточно проверенными программами. Кроме того, такие данные могут использоваться для аттестации адекватности имитаторов некоторых объектов внешней среды. Они могут быть полезны в тех случаях, когда создание определенных условий динамического функционирования объектов внешней среды очень дорого или опасно и может быть выполнено только в исключительных случаях.

В отличие от натурного эксперимента ***моделирование внешней среды*** и динамических тестов на ЭВМ имеет большие возможности контроля**,** как исходных данных, так и всех промежуточных и выходных результатов функционирования тестируемого программного комплекса или компонента. В реальных системах ряд компонентов иногда оказывается недоступным для контроля их состояния, так как либо невозможно поместить измерители контролируемых сигналов в

реальные подсистемы, подлежащие тестированию, либо это сопряжено с изменением характеристик самого анализируемого объекта. Преимуществом моделирования внешней среды на ЭВМ является также повторяемость результатов функционирования и возможность исследования большого количества вариантов и сценариев тестирования. В отличие от этого натурные эксперименты зачастую невозможно остановить на некоторой промежуточной фазе или повторить с абсолютно теми же исходными данными.

Программная имитация динамических тестов внешней среды на ЭВМ в реальном времени позволяет:

- проводить длительное непрерывное генерирование имитируемых эталонных тестов для определения характеристик функционирования комплекса программ в широком диапазоне изменения условий и параметров, что зачастую невозможно при использовании реальных объектов;
- расширять диапазоны характеристик имитируемых объектов за пределы реально существующих или доступных источников данных, а также генерировать динамические потоки информации, отражающие перспективные характеристики создаваемых информационных систем и объектов внешней среды;
- создавать тестовые эталоны, соответствующие критическим или опасным ситуациям функционирования объектов внешней среды, которые невозможно или рискованно реализовать при натурных экспериментах;
- обеспечивать высокую повторяемость имитируемых тестов при заданных условиях их генерации и возможность прекращения или приостановки имитации на любых фазах моделирования внешней среды.

Формализация документов как эталонов тестов комплексов программ

Эксплуатационная документация должна обеспечивать ***эффективное применение программного продукта*** и точно отражать его назначение, функции, характеристики и требования, для использования квалифицированными специалистами-пользователями. Для этого эксплуатационные документы необходимо тестировать на полное соответствие выполнения всей ***совокупности требований и выбранных эталонов на программный комплекс***, согласованных между менеджером

разработчиков и тестировщиками. В результате эксплуатационные документы можно использовать также как отдельный, независимый при разработке ***эталон и тип тестов*** для квалификационного тестирования реализации требований к функциям и характеристикам программного комплекса. Апробирование документов пользователями при применении программ является ***практическим методом тестирования*** корректности реализации требований к программному комплексу. Разработчики этих документов должны обеспечивать комфортное и корректное применение программных продуктов пользователями на основе ясного и непротиворечивого изложения в документах технологических процедур и операций для штатного применения, функционирования и получения требуемых результатов в версиях комплексов программ.

Эксплуатационная документация должна обеспечивать ***отчуждаемость программного продукта*** от их первичных поставщиков – разработчиков и возможность освоения и эффективного применения комплексов программ достаточно квалифицированными специалистами – пользователями. Эксплуатационные документы должны исключать возможность некорректного использования продуктов за пределами условий эксплуатации, при которых документами гарантируются требуемые показатели качества их функционирования и применения. Основная ее задача состоит в фиксировании, полноценном использовании и обобщении результатов функционировании программных продуктов и системы.

Организация документирования должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения эксплуатационных документов на программный комплекс и компоненты. Для этого в общем случае должны быть выделены ***специалисты***, которые обязаны планировать, утверждать, выпускать, распространять и сопровождать комплекты апробированных и утвержденных эксплуатационных документов. Они должны стимулировать разработчиков программных компонентов и комплекса, осуществлять непрерывное, регламентированное документирование процессов и результатов своей деятельности, а также контролировать полноту и качество утвержденных эксплуатационных и отчетных документов.

Технологическая документация процессов производства и обеспечения всего жизненного цикла, включающая подробные технические описания, и подготавливаемая ***как эталон*** для специа-

листов, ведущих проектирование, разработку и сопровождение комплексов программ, обеспечивает возможность их ***отчуждения***, детального освоения, развития и корректировки программ и данных. Технологическая документация непосредственно и в наибольшей степени должна отражать производственные процессы и результаты жизненного цикла комплексов программ и данных и требования к ним. Стандарты и нормативные документы должны регламентировать структуру, состав производственных этапов, работ и документов. Они должны: формализовать выполнение и документирование конкретных работ при проектировании, разработке и сопровождении комплекса программ; обеспечивать адаптацию документов к характеристикам среды разработки, внешней и операционной системы; регламентировать процессы обеспечения качества комплекса и его компонентов, методы и средства их достижения, реальные значения достигнутых показателей качества. Для контроля возможных изменений целесообразно предусматривать и согласовывать с заказчиком специальный документ, регламентирующий правила применения и корректировки их номенклатуры, а также состава и содержания документации, поддерживающей производство комплексов программ.

Технологическая документация программных комплексов и компонентов должна обеспечивать поддержку первичной инсталляции, безопасного функционирования и восстановления программ и данных после сбоев. Каждый из документов управления программным комплексом должен не противоречить ***международным стандартам*** на коммуникацию, интерфейсы с пользователями и базами данных, на защиту и обеспечение безопасности информации. Программные продукты и системы, поставляемые заказчикам или на рынок должны иметь ***конкретные, гарантированные и документированные цели, функции и характеристики***. В то же время в жизненном цикле комплексов программ невозможно в начале проекта предусмотреть все требования к функциям, характеристикам и качеству абсолютно точно, без дефектов и ошибок. Кроме того, сложные программные продукты обычно имеют длительный жизненный цикл с рядом версий, в течение которого они совершенствуются, расширяются, и повышается их качество. Поэтому процессы формирования изменений и реализации требований, тестов и эксплуатационных документов должны выполняться координировано и обеспечивать их взаимную адекватность. Их следует организовывать на ***основе мето-***

дологии регулируемых итераций множества версий, в которых периоды поставляемых, стабильных, испытанных версий комплекса программ, чередуются с интервалами пошагового расширения функций, совершенствования качества и оперативного адаптирования к характеристикам внешней среды у пользователей или в системах. Для этого в сложных проектах целесообразно применять международные стандарты, регламентирующие жизненный цикл программных компонентов и ***жесткую дисциплину*** координируемой деятельности коллектива специалистов для сопровождения и документируемого ***конфигурационного управления эталонными версиями*** требований, тестов и документов для обеспечения качества реализации изменяющихся требований.

Стандарты рекомендуют ***генерировать большое количество документов***. ***Ценность документации как эталона*** определяется, в частности, тем, насколько она помогает в организации процесса тестирования и поиске дефектов и ошибок комплекса программ. Хорошая документация должна обладать следующими преимуществами:

- облегчать тестирование и применение других типов эталонов;
- помогать организовывать взаимодействие между специалистами при одновременном тестировании компонентов и комплекса программ;
- представлять удобную структуру для организации, планирования и управления тестовыми процессами и эталонами.

Базой ***эффективного управления документированием*** должен быть ***план***, в котором задачи исполнителей частных работ согласованы с выделяемыми для них ресурсами, а также между собой по результатам и срокам их достижения. Реализация плана зависит от результатов выполнения производственных работ и документов и может требовать оперативной корректировки плана. Для этого необходимо следить за ходом проекта и документирования на всем протяжении жизненного цикла и сравнивать запланированные и фактические результаты работ и документы. ***Контроль адекватности и качества эталонных документов является органической функцией управления проектом*** и должен иметь средства регулирования поведения отдельных специалистов и коллектива разработчиков в целом. В плане управления документированием и обеспечением качества внимание специалистов должно акцентироваться на анализе достигнутых результатов разработки, методах и средствах достижения за-

данных характеристик комплекса программ. При планировании и разработке комплекс документации должен проверяться и аттестоваться на выполнимость и полноту в условиях ограниченных ресурсов, а также на корректность, адекватность и непротиворечивость отдельных документов.

Сложность, количество и полнота содержания набора документов, в первую очередь, зависят от ***масштаба – размера программного комплекса***, что целесообразно оценивать в начале проектирования. Для решения этой задачи необходимо детально исследовать требуемые ресурсы современных процессов создания, документирования и использования программ различных классов и назначения – встроенных, коммерческих, административных, учебных, уникальных. Состав и шаблоны документов целесообразно базировать ***на серии международных стандартов***, многие из которых адаптированы как ГОСТ Р (см. Приложение 1). Эти стандарты охватывают жизненный цикл систем и программных комплексов, а также их документооборот. Стороны – участники проекта программного комплекса, ответственны за выбор экономически эффективной модели состава и структуры документации для конкретного проекта и за адаптацию производственных процессов и шаблонов документов применительно к выбранной модели документооборота проекта программного комплекса.

Особое внимание в последнее время уделяется ***совершенствованию и детализации документов, обеспечивающих высокое качество создаваемых программных комплексов***, а также возможности их эффективного итерационного развития длительное время в многочисленных версиях. Соответственно должны изменяться документы, отражающие состояние производственных процессов и компонентов программных проектов. Для хранения, тиражирования и распространения документов сложных комплексов программ высокого качества целесообразно выделять группу специалистов, ответственных за ***контроль, обеспечение и гарантированное сохранение документации***. Для критических, важных систем документация на программы и данные должна храниться и дублироваться на различных типах носителей и эпизодически выводиться на бумажные носители. При определении схемы обеспечения сохранности документации разного содержания следует учитывать ее важность, трудоемкость хранения и возможность аварийного восстановления. Кроме то-

го, должна быть организована служба контроля, ответственная за соблюдение требований стандартов, нормативных и руководящих документов при подготовке документации всеми специалистами, участвующими в крупном проекте. Эта служба обязана обеспечить унификацию и высокое качество содержания, структуры и оформления документов.

Управление изменениями требований к комплексам программ

Управление изменениями требований – одна из ключевых задач итерационного формирования требований к комплексам программ. Функции и характеристики сложных комплексов программ обычно ***изменяются в течение всего жизненного цикла. Значения требований*** в начале разработки почти всегда отличаются от фактически достигнутых при завершении проекта и поставляемого программного продукта. Кроме того, показатели качества могут быть субъективными и отражать различные точки зрения специалистов. Необходимо уметь определять и примирять эти различные представления требуемого качества и их изменения, чтобы управлять ими должным образом на каждом этапе жизненного цикла (см. рис. 1.8).

Итеративная природа работы с требованиями определяет процессы проектирования и разработки версий программных комплексов. Кроме того, требования часто меняются в силу изменений систем и внешней среды, для которых они создаются и в которых эксплуатируется версия программного комплекса. Необходимо учитывать неизбежность изменений и планировать процессы по уменьшению проблем, связанных с ними. В то же время, не рационально рассматривать изменение требований в отрыве от других процессов. Далее изменения требований рассматриваются как ***автономные объекты*** сопровождения и управления конфигурацией. При этом делается акцент на итерационное управление изменениями требований, однако подразумевается, что за ними непосредственно ***следует адекватная их реализация*** в программном комплексе. В соответствии с требованиями заказчика и/или пользователей по развитию и модификации программного комплекса в его жизненном цикле должен быть организован процесс сопровождения требований. После поставки заказчику версии программного комплекса в соответствии с договором и предложениями о модификации или отчетами о дефектах разработ-

чик должен изменять требования к соответствующим программам и документам.

В техническом задании и в договоре с заказчиком следует четко определить ***порядок квалификации видов и причин изменений требований*** в программах и данных, а также ***распределение ответственности*** за их инициализацию, реализацию и финансирование. Это позволит избегать конфликтов между заказчиком, пользователями и разработчиками изменений в процессе последовательного, длительного развития версий программного комплекса. В соответствии с соглашением о распределении ответственности за изменения требований программ и данных должно осуществляться финансирование их реализации. Выявленные ошибки в требованиях, программах и данных, которые искажают реализацию основных функций, согласованных в договоре с заказчиком и отраженных в документации на версию комплекса программ, должны устраняться за счет разработчика. Если возможные корректировки документации не отражаются на применении комплекса программ пользователями, оформлять новую версию может быть не всегда целесообразно. Модификацию и расширение требований и функций компонентов или версии комплекса программ, ранее не отраженных в техническом задании, спецификации требований и договоре с заказчиком, следует квалифицировать как дополнительную работу с соответствующим финансированием заказчиком. При этом значительные изменения и расширения функций, вызывающие необходимость существенной корректировки документации для пользователей, целесообразно оформлять в составе новой версии программного продукта.

Корректирующее изменение требований может быть связано с необходимостью устранения фактических ошибок и дефектов в программном комплексе. Корректирующее сопровождение проводят в случае выявления несоответствия программного комплекса установленным требованиям. Этот вид изменений по причинам и источникам является непредсказуемым и его трудно планировать и регламентировать. При групповой работе специалистов, кроме личных ошибок программистов в модулях, они могут быть обусловлены некорректной интеграцией компонентов программ при наличии нескольких версий модулей, применением дублирующих или конфликтующих друг с другом копий модулей и различными дефектами взаимодействия программных компонентов между собой и с данными.

Поэтому изменения требований, обусловленные ошибками и выявленными дефектами, в большинстве случаев целесообразно накапливать и реализовывать, приурочивая к изменениям, регламентированным модернизациями версий.

Изменения при модернизации требований вносятся сверх технических и функциональных требований исходных спецификаций, установленных при первичном проектировании или выпуске предыдущей версии программного комплекса. Изменения, вносимые при адаптивном сопровождении, могут быть связаны с необходимостью приспособления программного комплекса к изменившейся внешней среде или требований пользователей с реализацией новых требований к системному интерфейсу, к самой системе или техническим средствам. Изменения, вносимые при полной модернизации требований, могут улучшать функциональные характеристики комплекса программ и его сопровождаемость. Такие изменения могут приводить к предоставлению пользователям новых функциональных возможностей, пересмотру технологии применения документов или к изменению самих документов. Сопровождение требований программного комплекса, необходимое для изменения структуры или системы, вносится в существующую архитектуру в рамках ограничений, установленных первичной структурой комплекса программ.

Стоимость реализации изменения требований к сложному компоненту или комплексу программ может определяться договором между заказчиком и поставщиком по прецедентам аналогичных проектов с учетом изменяющейся конъюнктуры. В развитии и совершенствовании программ могут быть заинтересованы как пользователи (заказчики), так и разработчики (поставщики) в разной степени. От этого зависит возможное распределение между ними затрат на сопровождение требований и их доля финансирования корректировок в конечном продукте.

Возможные затраты на развитие и совершенствование требований к версии комплекса программ зависят, прежде всего, от запросов и потребностей пользователей на новые функции и от готовности заказчика и разработчика удовлетворить эти потребности. ***Величина этих затрат определяется рыночной конъюнктурой для данного программного продукта*** и коммерческой целесообразностью его модификации и развития. По размеру вновь вводимых в очередную версию функций и компонентов с учетом степени новизны требований

для их разработки могут быть оценены затраты на реализацию. Организация и качество конфигурационного управления, а также структура и интерфейсы конкретного проекта существенно влияют на эту долю затрат при сопровождении. Дополнительные затраты могут потребоваться на комплексирование и испытания новых компонентов в составе очередной версии программного комплекса, а также на доработку документации.

Целью планирования изменений требований является подготовка плана работ по модификации и обеспечение ресурсов, необходимых для проведения этих работ после передачи версии программного комплекса на сопровождение. Этот план должен быть подготовлен разработчиками во время первичной разработки версии программного комплекса и включать в себя процессы итерационного анализа предложений заказчика по последовательному изменению требований. ***Общий план изменения требований должен определять***:

- анализ предложений о модификации требований и отчетов о их дефектах;
- причины необходимости изменения требований;
- разработку вариантов реализации изменения требований;
- состав исполнителей работ по разработке изменений требований;
- роли и обязанности каждого субъекта, вовлеченного в разработку результатов изменений требований;
- какие имеются и необходимы ресурсы для разработки результатов изменений требований;
- документально оформленные предложения о модификации требований и отчеты о дефектах, результаты их рассмотрения и варианты реализации изменений;
- методы и средства организации работ по конфигурационному управлению изменениями, выпуску версии комплекса и синхронизации отдельных работ;
- перечень всех проектных результатов анализа изменений требований и продуктов, подлежащих поставке заказчику;
- состав отчетных материалов по этапам, затратам и графикам проведения изменения требований;
- состав отчетных материалов по проблемам, модификациям и устраненным дефектам;

- время начала и длительность очередного изменения требований;
- выбранный вариант реализации изменения требований, согласованный с заказчиком и его утверждение.

Совет руководителей проектом по управлению изменениями требований к комплексу программ должен решать, какие изменения следует реализовать, при этом:

- содержимое базы данных предлагаемых изменений должно быть доступным для анализа всеми заинтересованными в проекте лицами;
- первоначальный текст запроса на изменение должен анализироваться, не изменяться или не удаляться;
- анализ влияния изменения на проект и пользователей необходимо выполнять для каждого изменения;
- каждое предлагаемое для реализации изменение требования следует отслеживать вплоть до утверждения этого изменения;
- обоснование каждого утверждения или отклонения запроса на изменение необходимо документировать и сохранять.

На практике можно делегировать принятие некоторых детальных решений, касающихся требований разработчикам, однако ни одно соглашение, влияющее на работу системы, не должно идти в обход процесса контроля и утверждения изменений. Процесс изменений требований должен иметь ***быстрый путь***, чтобы ускорять не рискованные и дешевые изменения в сжатом цикле принятия и реализации решений. Запрос на изменение требований должен проходить через определенные этапы на протяжении жизненного цикла комплекса программ, причем на каждом он имеет ***различные статусы и права*** (см. таблицу 1). Группа специалистов, которая подготавливает решение, может утвердить или отклонить каждое предложенное изменение для определенного компонента или всего комплекса. Руководитель проекта или менеджер, возглавляющий Совет по управлению изменениями, обладает правом принятия окончательного решения на утверждение крупных изменений версии комплекса программ.

Стандартом **ISO 14764** рекомендуется формализовать и выделять ***конкретный план изменений требований к программному комплексу*** из общего состава процессов жизненного цикла, который уточнять и адаптировать с учетом размера и особенностей проекта. Персонал сопровождения требований должен устанавливать проце-

дуры для получения, записи и сообщений о дефектах, проблемах и запросах на изменения требований от заказчика и пользователей и для обеспечения обратной связи к ним. Следует устанавливать функциональную ***связь с процессом управления конфигурацией*** для управления модификациями в существующем программном комплексе.

Организация изменений и сопровождения требований к комплексам программ

Стратегия сопровождения требований должна быть ориентирована на людские и материальные ресурсы, необходимые и доступные для обеспечения развития и модификаций программного продукта. Этот анализ является исходным, и должен быть определен в общей стратегии сопровождения программного продукта. Анализ дефектов и модификаций требований включает оценку их влияния на организацию, существующую систему и интерфейсы системы для:

- корректировок, усовершенствований, профилактики или адаптации программного продукта к новой среде;
- определения размера модификации, стоимости, времени на модификацию комплекса программ;
- оценки риска влияния на производительность, надежность или безопасность программного комплекса.

Менеджеру изменения требований следует определять процедуры для получения, документирования и контроля отчетов о дефектах и предложений о модификациях от заказчика и пользователей, а также для обеспечения обратной связи с пользователями. Каждая возникающая проблема (и/или дефект) должна быть документально оформлена и введена в ***процесс анализа изменений***, для чего следует:

- разработать схему классификации и присвоения приоритетов для предложенных модификаций требований и описаний дефектов;
- разработать процедуры анализов рентабельности изменений требований;
- определить процедуры и формы представления предложенных модификаций и описаний дефектов заинтересованными лицами;
- определить организацию обратной связи с заказчиком и пользователями при анализе изменений;
- определить, как пользователей будут обслуживать в период реализации новых требований в версии программного комплекса;

- определить, как будут введены предлагаемые модификации в базу данных учета конфигурации изменений требований и используемых ресурсов.

Процесс контроля изменений требований должен позволять руководителю проекта принимать решение, которое удовлетворяет заказчика и имеет коммерческую ценность, при этом должны учитываться затраты на жизненный цикл комплекса. Следует отслеживать статус всех предложенных изменений и убедиться, что предложенные требования не были потеряны или упущены. Руководство проектом должно довести до сведения сотрудников политику, в которой описываются ожидания того, как участники проекта должны обрабатывать предложенные изменения требований.

Для сохранения и повышения качества сложных комплексов программ ***необходимо регламентировать процессы модификации и совершенствования требований***, а также поддерживать их контролем качества. Технология сопровождения требований должна обеспечивать ***итерации*** координированного развития множества версий программного комплекса и их компонентов, каждая из которых имеет достаточно высокое качество и специфические функции, а также, возможно, различных пользователей. Благодаря этому со временем программные комплексы должны улучшаться и совершенствоваться как по функциональным возможностям, так и по качеству решения каждой задачи.

Сопровождаемость – возможность регламентированной модификации комплекса программ является важной характеристикой, для заказчиков, поставщиков и пользователей, отражающей возможность и простоту внесения изменений в требования и в программный комплекс после его ввода в эксплуатацию (стандарт **ISO 14764**). Эти требования количественно можно устанавливать как изменяемость и тестируемость экономическими категориями ***допустимой трудоемкости и длительности реализации*** этих задач при некоторых средних условиях. Требуемые значения зависят от четкости концепции и архитектуры комплекса программ, от унифицированности внутренних, внешних и с пользователями интерфейсов, от качества технологической документации, а также от инструментальной оснащенности. Обобщенно это отражается на длительности и трудоемкости подготовки и реализации типовых изменений, обусловленных необходимостью устранения дефектов и усовершенствованиями функций ком-

плекса программ. Для подготовки и выполнения каждого изменения (без учета затрат времени на обнаружение и локализацию дефекта) нужно устанавливать допустимую среднюю продолжительность и суммарную трудоемкость работ специалистов при их реализации.

Сопровождаемость требований должна быть определена до начала первичной разработки проекта комплекса программ соответствующим ***соглашением между заказчиком и разработчиком - поставщиком***. Разработчик должен подготовить план обеспечения сопровождения требований, в котором отражены конкретные методы, соответствующие ресурсы и последовательности работ. Следует определить необходимые затраты по обеспечению мониторинга и оценки сопровождаемости в процессе разработки. Требования к процессам сопровождения определяются группой основных факторов, влияющих на реализацию модификации программных комплексов, образующих концептуальную цепочку: ***требования на изменения – изменяемые функции – размер (масштаб) изменений – стратегия модификаций – ресурсы, необходимые для их реализации.*** Эта логическая схема обычно используется при последовательном анализе процессов сопровождения сложных комплексов программ. При этом основным критерием оценки сопровождения требований является совершенствование функциональной пригодности и улучшение характеристик качества программного комплекса.

Основной процесс эксплуатации в жизненном цикле может инициировать процесс сопровождения требований программного комплекса путем представления предложений о модификации (изменении) или отчетов о дефектах. Процесс сопровождения программного продукта в соответствии со стандартом **ISO 12207** и детализацией этого раздела в стандарте **ISO 14764** должен использовать основной ***стандартизированный процесс*** разработки комплексов программ и вспомогательные процессы документирования, управления конфигурацией, обеспечения качества, верификации, аттестации, совместного анализа, аудита и устранения дефектов. Организационные процессы управления, создания инфраструктуры и обучения должны определяться менеджером в начале сопровождения требований.

Если заказчику необходимо вести сопровождение требований силами разработчика после поставки данного программного комплекса или по окончании гарантийного периода, это должно быть указано в соответствующем соглашении. Поставка модернизирован-

ных документов должна быть предусмотрена в соответствующем соглашении. Также должно быть оговорено обучение персонала пользователей. Поставщик должен подготовить процедуры выполнения каждой задачи сопровождения, выполнять эти процедуры во время сопровождения и проверять соответствие конкретных работ договорным требованиям и установленным процедурам.

Сопроводитель и заказчик должны ***заключить договор на сопровождение требований*** и указать в нем возможные процедуры внесения изменений в сопровождаемые программные комплексы (см. рис. 1.8). Процедуры могут быть использованы как разработчиком оригинала версии программного комплекса, так и независимым сопроводителем и ***охватывать***:

- основные требования и правила, используемые для определения того, когда комплекс программ может быть локально откорректирован, а когда необходима новая версия с использованием для ее подготовки и инсталляции процесса разработки;
- описания типов редакций версий в зависимости от частоты их появления или влияния на эксплуатацию программного комплекса (например, экстренные редакции, периодические редакции);
- способы информирования заказчика о состояниях текущих или намечаемых изменений требований;
- классификацию типов изменений, их очередности (приоритетности) реализации и взаимосвязи с другими предложенными изменениями.

При ***подготовке процесса изменения требований*** следует создать планы и определить процедуры, выполняемые при реализации сопровождения. План сопровождения требований целесообразно создавать параллельно с планом разработки первой версии программного продукта. При выполнении данной работы сопроводитель также должен определить необходимые организационные интерфейсы и взаимосвязи между специалистами и с другими предприятиями. Для обеспечения эффективной реализации процесса сопровождения следует разработать и документально оформить ***стратегию проведения сопровождения требований***. При реализации этой деятельности необходимо: разработать планы и процедуры сопровождения; установить процедуры рассмотрения предложений о модификации и отчетов о дефектах; применить управление конфигурацией. Сопроводитель должен разработать, документально оформить и выполнить планы и

процедуры для проведения работ и решения задач процесса сопровождения требований.

Спецификация требований на изменения программного комплекса должна исчерпывающе и однозначно описывать обязательные требования к комплексу и к его модификациям и отражать характеристики качества, требуемые стандартами. При этом должны быть учтены следующие ***факторы, влияющие на сопровождаемость требований***:

- определение и описание новых функций;
- точность и логическая организация данных;
- интерфейсы (системные, машинные и пользователей), особенно новые и перспективные интерфейсы;
- требования к функциям и рабочим характеристикам, включая влияния корректировок и дополнений;
- требования, налагаемые запланированной внешней средой;
- неоднородность требований, определяющая простоту или сложность их трассировки и прослеживания;
- план обеспечения качества модифицированного программного продукта, в котором особое внимание должно быть уделено документам на изменения и их согласованность.

Целенаправленная и методичная экспертная ***оценка возможного масштаба и ресурсов*** на изменения уменьшает величину ошибок, однако обычно она остается все-таки довольно большой. Для обеспечения достоверности первичное прогнозирование целесообразно проводить путем экстраполяции на базе накопленных конкретных данных об отдельных аналогичных модификациях версий программного комплекса. Важной частью концепции сопровождения требований и их реализацией является ***определение ресурсов и специалистов*** (физических или юридических), отвечающих за сопровождение. Это в равной степени справедливо и в случае внутреннего сопровождения в производственной организации. При выполнении сопровождения по соглашению с третьей стороной это должно быть отмечено в концепции. ***Выбор предприятия сопроводителя*** должен быть основан на анализе следующих ***факторов***:

- возможный срок службы программного комплекса;
- размер допустимых первичных и долгосрочных затрат на сопровождение;
- необходимая квалификация сопровождающего персонала;

- функциональная пригодность и работоспособность исходной версии программного комплекса;
- программа (график) модификаций и сопровождения;
- уровень квалификации сопроводителя в предметной области применения программного комплекса.

Должна быть проведена ***оценка условий финансирования и стоимости сопровождения требований***. Стоимость зависит от размера возможной области изменения характеристик программного комплекса при сопровождении. Дополнительными факторами являются стоимости: обучения как сопроводителей, так и пользователей; среды программного инструментария, средств тестирования и их сопровождения. При первоначальном определении требований к функциональной пригодности и к конструктивным характеристикам заданные заказчиком ***ограничения ресурсов*** не всегда могут учитывать ряд особенностей сопровождения, что обусловит недопустимое снижение (или завышение) требований к некоторым характеристикам модифицированного программного комплекса. Кроме того, возможно, что некоторые характеристики или их изменения противоречивы или принципиально нереализуемы в данном проекте. В результате ***не сбалансированные изменения требования*** и доступные ресурсы проявятся в виде потерь в качестве или в потребности выделения дополнительных ресурсов.

В зависимости от сложности проекта окончательным результатом работ при ***прогнозировании изменений комплекса программ*** должны быть детализированные и утвержденные требования к номенклатуре, свойствам и значениям качества версии программного комплекса, которые достаточны для его полноценного сопровождения и последующей эффективной эксплуатации. Эти ***требования закрепляются в контракте и техническом задании***, по которым сопроводитель впоследствии должен отчитываться перед заказчиком при завершении модификаций. Принципиальные и технические возможности, точность реализации свойств и измерения значений характеристик программного комплекса, а также общие ресурсы конкретного проекта всегда ограничены в соответствии с их содержанием и возможностями заказчика и разработчиков. Это определяет ***рациональные диапазоны значений каждого изменения***, которые могут быть выбраны в концепции сопровождения на основе требований заказчика, здравого смысла, а также путем анализа пилотных проектов

и прецедентов в спецификациях требований реализованных модификаций.

Сопроводитель должен согласовать и получить подтверждение того, что ***внесенные изменения удовлетворяют требованиям заказчика***, установленным в договоре: посредством вспомогательного процесса обеспечения качества; испытаний выполнения этого процесса; проведения аудита функциональной и физической конфигурации. Результатами данной работы являются: новая версия, включающая в себя принятые изменения требований; отклоненные изменения; отчет о приемке версии; отчеты о проверках и аудитах; отчет о квалификационном тестировании и испытаниях. Требования к функциональным характеристикам и качеству, ***утвержденные после изменений требований***, могут быть закреплены в техническом задании как обязательные для детального и рабочего проектирования модификации. Эти данные могут использоваться при последующем оценивании качества и при их сопоставлении с требованиями в процессе квалификационных испытаний, сертификации модификаций или новой базовой версии программного комплекса.

При детальном проектировании изменений может быть целесообразен ***контроль реализации договора*** и уточнение совокупности функциональных характеристик и качества модифицированных версий, а также соотношения качества и затрат ресурсов на изменения комплекса программ. Для заказчика и пользователей при сопровождении может иметь значение не только определение функциональной пригодности, но и оценка потенциального спроса на рынке конкретного комплекса, а также его ***конкурентоспособности*** с другими аналогичными по функциям программными комплексами с учетом его качества и стоимости. Это обстоятельство может определять необходимость контроля реализации и ***уточнения требований*** к отдельным характеристикам для оценивания интегрального качества новой версии, поставляемой на рынок.

Лекция 1.7

ВЕРИФИКАЦИЯ, ТРАССИРОВАНИЕ И ОБЕСПЕЧЕНИЕ БАЛАНСА ТРЕБОВАНИЙ К КОМПЛЕКСАМ ПРОГРАММ

Верификация качества требований к комплексам программ

Широко распространенная ***неупорядоченная разработка требований*** к функциям, процедурам и взаимодействию программных компонентов не может гарантировать высокое качество сложных программных комплексов. При этом пропускаются и не проверяются некоторые сценарии и маршруты исполнения программ, которые реализуют важнейшие требования контракта и исходных спецификаций. Удовлетворение потребностей заказчика является целью любого программного проекта. Соответственно, ***обеспечение качества, полноты и корректности требований*** в проекте невозможно представить без адекватных процессов работы с ними – начиная с ***верификации требований***, заканчивая проверкой соответствия получаемого программного комплекса этим требованиям на всех этапах его создания (рис. 1.9)

Верификация – это действия по проверке выполнения на ***n-ом*** этапе процесса разработки того, что было запланировано на ***(n-1)-ом*** этапе. Во время создания архитектуры системы и при определении требований специалист по тестированию должен сравнивать обусловленные архитектурой ресурсы с общими требованиями. Во время тестирования рабочей проектной документации тестировщик должен сверять эту документацию с документацией по архитектуре, которая была разработана на этапе высокоуровневого проектирования. Постоянно выполняемый процесс проверки того, что каждый шаг разработки является корректным, удовлетворяет ***потребности последующей деятельности*** и не является излишним, называется ***верификацией***.

Верификация, трассирование и обеспечение баланса требований к комплексам программ должны включать:

- верификацию качества требований к комплексам программ:
  - верификацию договора с заказчиком на цели и функции системы с комплексом программ;
  - верификацию требований к функциям системы;
  - верификацию комплекса, компонентов и модулей программ;
  - верификацию результатов интеграции компонентов и модулей в комплекс программ;
  - верификацию эксплуатационной и технологической документации на компоненты и комплекс программ;
- трассировку требований к взаимодействию функций и компонентов комплекса программ:
  - направления трассировки: к требованиям компонентов и от требований к компонентам;
  - явную и неявную трассировку требований к компонентам;
  - корректную матрицу трассировки требований к компонентам;
  - контроль последовательности действий для реализации трассировки требований;
- обеспечение баланса требований к качеству компонентов комплекса программ с учетом рисков и доступных ресурсов:
  - поэтапную разработку и уточнение спецификаций требований к комплексам программ;
  - уточнение распределения конкретных требований и ресурсов по критериям: максимизации функциональной пригодности или минимизации рисков применения;
  - выделение приоритетных требований и характеристик к компонентам и комплексу программ;
  - сравнительный анализ приоритетов требований и допустимых рисков для выравнивания уровня требований.

Рис. 1.9

Стандарт **IEEE 1012** определяет верификацию как ***процесс последовательного оценивания системы, комплекса, компонента или модуля с целью определить, удовлетворяют ли их результаты условиям, наложенным на предыдущем этапе разработки.***

Таким образом, верификация в значительной мере является аналитической деятельностью, в ходе которой необходимо убедиться, что каждая стадия разработки (например, реализация в программном комплексе одного или нескольких требований) соответствует ***требованиям, заданным на предыдущей стадии***:

- описанные функции действительно соответствуют потребностям заказчика и пользователей;
- производные от этих функций требования действительно поддерживают заданные функции;
- проектирование поддерживает функциональные и нефункциональные требования поведения системы;
- тексты и содержание программных компонентов и модулей действительно соответствует результатам и целям проектирования комплекса программ;
- тесты обеспечивают полное покрытие разработанных требований к компонентам, модулям и содержанию текстов программ.

Для этого нужен план верификации и автоматизированные средства, которые помогут выполнить его. В команде разработчиков необходимо понять и усвоить, что такое верификация, и взять на себя ответственность за ее проведение. Процесс анализа требований т – трансформации информации, полученной от пользователей и других заинтересованных лиц в четко и однозначно определенные требования, передаваемые разработчикам для реализации в компонентах и комплексе программ, ***включает***:

- обнаружение и разрешение конфликтов между требованиями;
- определение границ функциональной задачи, решаемой создаваемым программным комплексом, в общем случае – определение концепции и содержания программного проекта;
- детализация требований к системе для установления требований к программному комплексу, его компонентам и модулям.

Поэтому в стандартах, регламентирующих жизненный цикл сложных комплексов программ, значительное внимание уделяется процессам ***упорядоченной, иерархической верификации***, исходящей от требований к системе и к комплексу программ. Ее основой является последовательная ***детализация и трассировка требований*** к программным компонентам разных уровней и контроль их согласованной реализации в жизненном цикле комплекса программ.

Верификацию следует проводить постоянно, чтобы убедиться, что каждый компонент и модуль корректен, удовлетворяет потребности следующего компонента и не содержит лишних или неопределенных процедур. Важным показателем качества реализации компонента и комплекса программ является возможность трассировки требований на последовательных этапах спецификации, архитектуры, проектирования, реализации и тестирования. Способность отслеживать ***отношения между требованиями*** и учитывать их связи при возникновении изменений является основой многих современных высоконадежных программных процессов, особенно в критических областях создания и применения программного продукта.

Эффективным методом ***реализация корректной (правильной) системы, адекватной требованиям*** является использование прототипов требований и прецедентов их реализации в качестве основы архитектуры и реализации проекта. Необходимо идентифицировать все возможные источники требований, значимые для решения задач проекта. Только после этого можно верифицировать и определять их влияние на продукт. Следует оценивать степень полноты ***информированности источников требований*** и их значимость для учета влияния на цели проекта системы и комплекс программ:

- знание предметной области системы;
- состав и квалификацию заинтересованных лиц;
- операционное и внешнее окружение среды;
- организационную среду коллектива разработчиков.

Выделение приоритетов и однозначность требований, передаваемых специалистам – разработчикам, связь между требованиями и их взаимное влияние друг на друга – все это должно являться следствием четкого и однозначного понимания задач источников требований – ***заинтересованных лиц***. Хотя верификация является аналитическим методом, рекомендуется помнить о важности интуиции. Нельзя ограничиваться механическим применением верификации. Основное внимание при ее проведении следует уделять использованию трассировки для отбора функций и компонентов системы, нуждающихся в проверке их взаимодействия. ***Методы верификации и проверки корректности требований*** призваны гарантировать, что все проверки служат основной цели и функциям проекта и не являются избыточными.

Чтобы принять решение о том, какая часть системы нуждается в верификации и проверке корректности и в каком объеме, целесообразно применять ***анализ и оценку рисков***. Это позволяет определить, для каких компонентов требований их неправильная реализация недопустима, а также разработать план действий по верификации и проверке правильности, основываясь на результатах этих оценок. На этом этапе следует привлекать к управлению требованиями и анализу их корректности, специалистов по тестированию, подключая их к планированию тестов с самого начала проекта. Группа тестирования должна разрабатывать тестовые процедуры и сценарии, которые трассируются к прецедентам, а также функциональным и конструктивным требованиям.

Верификация проекта системы – это процесс для определения, выполняют ли программный продукт и его компоненты требования и/или условия, наложенные на них в предыдущих действиях. Для эффективности затрат и выполнения верификация должна быть ***интегрирована*** как можно раньше с процессами жизненного цикла системы (такими как поставка, разработка, финансирование или сопровождение), которые используют ее. В случае, когда процесс выполняется администрацией, независящей от поставщика, оператора, разработчика или персонала сопровождения, он составляет ***независимый процесс верификации***.

Если заказчик проекта требует ***гарантировать независимую верификацию*** всего комплекса требований, то должна быть выбрана квалифицированная организация, ответственная за выполнение полной верификации. Этой организации должны гарантироваться независимость и полномочия верификационной деятельности по обеспечению корректности совокупности требований. Должны быть определены плановые действия жизненного цикла системы и программные продукты, требующие верификации. План должен определять требуемые верификационные задачи для каждой деятельности жизненного цикла программного продукта и связанные ресурсы, обязательства и программы. Дефекты и несоответствия, обнаруженные при верификации, должны быть введены в ***процесс корректировки требований к комплексу программ***. Результаты верификационной деятельности должны быть доступны заказчику и потребителю системы и программного продукта.

Основные регламентированные задачи ***верификации проектов систем и программных комплексов*** отражены в стандарте **ISO 12207** и включают в свой состав следующие задачи и критерии (см. рис. 1.9):

- ***верификация договора с заказчиком на цели и функции проекта*** по критериям:
  - возможность поставщика удовлетворить установленным требованиям заказчика;
  - требования договора непротиворечивы и покрывают все потребности заказчика и пользователей;
  - предусмотрены адекватные процедуры для регулирования и выполнения изменений требований договора и устранения его дефектов;
  - предусмотрены процедуры для сотрудничества между участниками – сторонами проекта, включая право собственности, гарантию, авторское право и конфиденциальность;
  - предусмотрены критерии и процедуры приемки результатов разработки на соответствие требованиям договора
- ***верификация требований к функциям системы*** по критериям:
  - требования к системе непротиворечивы, выполнимы, тестируемы и проверяемы;
  - требования системы распределены между компонентами аппаратных, программных средств и ручными операциями полностью и в соответствии с функциями и критериями проекта;
  - требования к программному комплексу последовательны, выполнимы, тестируемы и точно отражают требования системы;
- ***верификация требований к модулям, компонентам и комплексу программ*** по критериям:
  - тексты программных модулей и компонентов удовлетворяют требованиям к комплексу программ, тестируемы, корректны и соответствуют стандартам программирования;
  - программа отражает требуемые события, последовательные интерфейсы и управляющие потоки, завершенность, использование временных и вычислительных ресурсов, определение ошибок, их обнаружение и восстановление работоспособности компонентов и комплекса программ;
  - комплекс программ корректно обеспечивает надежность, безопасность, производительность и другие критические требования;

- ***верификация интеграции*** модулей, компонентов и комплекса программ по критериям:
  - модули и компоненты полностью и правильно могут быть интегрированы в комплекс программ;
  - компоненты аппаратных средств, программного продукта и ручные операции полностью и правильно интегрированы в систему;
- ***верификация документации*** по критериям:
  - эксплуатационная и технологическая документация адекватна, корректна, полна и непротиворечива;
  - подготовка документации выполнена своевременно;
  - управление конфигурацией документов соответствует плановым процедурам производства.

После общей верификации требований к проекту системы, комплекса и компонентов программ необходима разработка и углубленная последовательная верификация требований к модулям и компонентам.

Верификация комплекса и компонентов программ – это процесс, предназначенный для определения, выполняют ли программный комплекс, его компоненты, модули, требования и условия, наложенные на них в предыдущих этапах жизненного цикла комплекса программ. Для эффективности затрат при ее реализации, верификация должна быть интегрирована как можно раньше с процессами проектирования и производства программ. Эти процессы включают анализ, трассирование и тестирование выполнения требований, которые являются важнейшими компонентами верификации. Они проводятся ***сверху вниз***, начиная от общих требований в техническом задании и/или спецификации на всю систему до детальных требований на отдельные компоненты, модули и их взаимодействие.

Назначение верификации комплекса программ состоит в том, чтобы обнаружить, зарегистрировать и устранить дефекты и ошибки, которые могли быть ***внесены в требования*** во время разработки или модификации программ. Основное назначение верификации проверить, что (см. рис. 1.9):

- общие требования к системе, предназначенные для программной реализации, корректно переработаны в требования высокого уровня к программному комплексу и удовлетворяют исходным системным требованиям;

- требования высокого уровня к комплексу программ правильно переработаны в архитектуру комплекса и в требования к функциональным компонентам низкого уровня и модулям, которые удовлетворяют требованиям высокого уровня;
- архитектура комплекса программ и требования к компонентам низкого уровня корректно переработаны в удовлетворяющий им исходный текст модулей программ.

Цели верификации требований к комплексу программ достигаются посредством ***последовательного выполнения комбинаций*** из трассировки, анализов, разработки тестовых сценариев и процедур, и возможности последующего выполнения этих процедур. Трассировки и анализы должны обеспечивать оценку точности, полноты и верифицируемости требований, архитектуры комплекса и исходного текста программ. Тестовые сценарии предназначены для проверки внутренней непротиворечивости и полноты реализации требований. Анализ тестовых процедур должен обеспечивать демонстрацию соответствия требований к программным модулям и компонентам, исходным требованиям к программному комплексу (рис. 1.10).

В стандартах на жизненный цикл при выполнении работ по верификации комплекса программ рекомендуется следующая ***последовательность основных процедур***:

- должны быть проверены корректность и непротиворечивость исходных требований высокого уровня (контракта) и продемонстрирована трассируемость и прослеживаемость к ним последующих требований;
- результаты анализа прослеживаемости и анализа покрытия исходных требований и структуры должны показывать, что каждое требование к комплексу программ ***является трассируемым*** к компонентам и объектному коду модулей, которые его реализуют, а выполненные просмотры, анализы и сгенерированные тестовые наборы способны проверить эти требования;
- когда невозможно проверить некоторые требования посредством исполнения программ в тестовой среде, адекватной реальной, должны быть обеспечены другие средства, и в документах должна быть обоснована возможность их использования для удовлетворения целей верификации комплекса программ;
- несоответствия и ошибки, обнаруженные в процессе верификации должны быть зарегистрированы для последующего исследова-

ния и корректировки компонентов, комплекса программ и/или требований.

Рис. 1.10

Верификация, просмотры и анализы требований высокого уровня предназначены обнаруживать, регистрировать и устранять ошибки, которые внесены в процессе разработки и преобразования требований к комплексу программ. Эти просмотры и анализы должны подтвердить корректность и согласованность требований высокого уровня, а также ***гарантировать***, что:

- полностью определены функции системы, которые должен выполнять программный комплекс, а также требования к характеристикам по функциональности, эффективности, надежности и безопасности системы, удовлетворены в исходных требованиях высокого уровня, правильно определены производные требования и обоснована их необходимость;

- каждое требование высокого уровня к программному комплексу является точным, однозначным и достаточно детализированным, требования не конфликтуют друг с другом;
- не существует конфликтов между требованиями высокого уровня и возможностями аппаратных и программных средств реализующей ЭВМ, особенно такими, как время реакции системы и аппаратура ввода/вывода;
- каждое требование высокого уровня к программному комплексу может быть проверено и верифицировано;
- процесс разработки требований к комплексу программ полностью соответствует стандартам на создание требований и обоснованы любые отклонения от стандартов;
- системные, функциональные требования, требования по эффективности, надежности и безопасности системы, предназначенные для программной реализации, отражены в требованиях высокого уровня.

Верификация, просмотры и анализы требований архитектуры программного комплекса предназначены обнаруживать и регистрировать дефекты и ошибки, которые могли быть внесены во время разработки архитектуры комплекса программ. Эти просмотры и анализы должны подтвердить, что архитектура комплекса программ не находится в противоречии с требованиями высокого уровня, выполняются все функции, которые гарантируют целостность системы, а также существуют корректные связи между функциональными компонентами архитектуры комплекса.

Верификация, просмотры и анализы требований к программным компонентам и модулям предназначены выявлять дефекты и ошибки детализированных требований, которые внесены в процессе проектирования функциональных компонентов. Эти анализы должны гарантировать, что требования низкого уровня к компонентам и модулям удовлетворяют требованиям высокого уровня и что правильно определены формируемые из них требования и обоснована их необходимость. Следует гарантировать, что каждое требование низкого уровня является точным, однозначным и достаточно детализированным ***для программирования модуля*** и что требования низкого уровня не конфликтуют друг с другом.

Верификация, просмотры и анализы исходного текста модулей программ должны выявлять и регистрировать ошибки, которые

внесены в процессе программирования модулей и компонентов комплекса. Они должны подтверждать, что выходные результаты программирования модулей являются точными, полными и могут быть верифицированы. Прежде всего, должна проверяться корректность текста программ по отношению к исходным требованиям к архитектуре комплекса программ. Анализы обычно ограничиваются исходным текстом модулей программ, проверяется, что он является корректным, полным и соответствует требованиям к соответствующим компонентам низкого уровня, а также то, что исходный текст не содержит не описанных функций. Должно быть проверено, что процесс разработки программ осуществлялся полностью в соответствии с ***верифицированными требованиями к компонентам и модулям***, а также ***по стандартам на программирование***, и отклонения от этих стандартов обоснованы, особенно в случаях ограничения сложности и использования конструкций программ, которые должны удовлетворять целям безопасности системы.

Верификация, просмотры и анализы тестовых вариантов, процедур и результатов должны показать, что требования к тестированию комплекса, компонентов и модулей разработаны и могут выполняться точно и полностью (см. рис. 1.10). Должно быть рассмотрено и подтверждено верификацией, что тестовые сценарии и варианты исходных данных правильно представлены в процедурах тестирования и ожидаемых результатах; гарантируют, что составы результатов тестирования требований являются корректными и что объяснимы расхождения между фактическими и ожидаемыми результатами.

Неприятным моментом верификации является то, что можно потратить время на верификационную деятельность, которая не принесет ожидаемой отдачи. Поэтому нужен способ предварительной ***оценки «экономической эффективности»*** верификационной деятельности. В той или иной степени оценкой затрат на верификацию приходится заниматься на ***каждой*** стадии разработки. Она необходима, чтобы удостовериться в правильности и затратах на переходы:

- от потребностей пользователя к функциям программного комплекса;
- от требований к комплексу программ к его архитектуре;
- от архитектуры к модели проектирования и производства

модулей, компонентов и комплекса программ;

- от реализации модулей, компонентов и комплекса программ к планированию и реализации совокупности тестов (см. рис. 1.10).

Особое внимание следует уделять этапу ***завершения проекта***. В успешном процессе это еще один шаг – испытания, подтверждающий, что система сконструирована и реализована в соответствии ***с потребностями заказчика***. Кроме того, на этом этапе необходимо показать, что система действительно работает так, как требовалось.

Трассирование требований к сложным комплексам программ

Поддержка верификации осуществляется путем использования методов ***трассировки требований***, что позволяет связывать друг с другом части проекта, проводить проверку адекватности требований к прецедентам их реализации и функциям, и обратно. С помощью трассировки можно удостоверяться в том, что:

- все компоненты исходных требований проекта учтены;
- все реализуемые компоненты служат заданной цели и требованиям к комплексу программ.

Посредством ***трассировки*** следует устанавливать корректность связей между двумя или большим числом ***компонентов и/или процессов разработки требований***, которые являются: предшествующими – последующими или главными – подчиненными, а также соответствие между требованиями и их реализацией конкретными программными компонентами. Каждый компонент и модуль программного комплекса должен ***оправдывать свое существование и соответствовать каким-то заданным требованиям***. Ключевыми элементами верификации и тестирования являются отношения трассировки. Эти отношения можно определять с помощью модели, использующей понятия ***«трассируется к»*** и/или ***«трассируется от»***. Если одно или несколько требований к программному компоненту создаются с целью поддержки некоторой функции, заданной в исходном документе, то требование трассируется ***от*** некоторой функции. Если некоторое требование к программному компоненту «трассируется ***к***» определенному тестовому сценарию, то данное требование тестируется этим скриптом. То, что описание компонента «трас-

сируется ***от***» конкретного программного требования, подразумевает, что это требование реализуется указанным компонентом.

Потребности заказчика должны отслеживаться путем ***анализа содержания требований***, чтобы можно было определить, какие требования будут затронуты, если в течение или после разработки, потребности изменятся. Это также дает уверенность, что в спецификации требований указаны все потребности заказчика. Кроме того, можно проследить ***в направлении от требований*** к потребностям заказчика, чтобы определить происхождение каждого требования к комплексу программ. Если необходимо представить потребности заказчика в форме сценариев использования функций, то анализ должен отражать ***трассирование между вариантами использования*** и функциональными требованиями.

По мере разработки комплекса программ, процессы можно отслеживать ***в направлении от требований***, и определять связи между отдельными требованиями и компонентами комплекса. Этот тип связей гарантирует, что каждое требование удовлетворено, поскольку установлено, какой компонент соответствует этому требованию. Еще один тип связей может контролировать отдельные элементы продукта в ***направлении к требованиям*** для того, чтобы знать ***причину и цель*** создания каждого компонента. В большинстве комплексов программ могут быть компоненты, не относящиеся напрямую к требованиям заказчика, но необходимо устанавливать, для чего нужен каждый компонент.

Если трассировщик обнаружит незапланированную функциональность при отсутствии соответствующего требования, то фрагмент программы может свидетельствовать, что разработчик реализовал требование, которое аналитик или заказчик может добавить к спецификации. Однако это может быть элемент программы, ***«украшающий»*** фрагмент, который не относится к комплексу. Связи трассируемости помогут сортировать подобные ситуации и получать более полное представление о том, как именно компоненты системы ***составляют целое, соответствующее требованиям***. Сценарии верификации или тестирования, которые созданы на основе отдельных требований, которые можно проследить до этих требований, представляют собой механизм выявления нереализованных требований, поскольку нет ожидаемой функции или компонента. Не всегда необходимо определять и управлять всеми типами связей трассируемости.

Во многих проектах удается получать 80% эффекта от применения трассируемости для выявления дефектов, вложив приблизительно 20% усилий от требующихся для полного покрытия контролем заданных требований. Если нужно отслеживать тестирование системы только до функциональных требований или сценариев использования, то следует решать, какие связи уместно контролировать, и это в значительной степени будет способствовать успешной разработке и эффективной модификации комплекса. Пропуск реализации, верификации и тестирования требования может быть существенным дефектом, если заказчик не удовлетворен или в готовом продукте отсутствует функция, особо важная для обеспечения надежности или безопасности.

Трассирование требований сложного комплекса программ – трудоемкая задача, обычно выполняемая вручную, для которой необходима соответствующая организация и квалификация специалистов. Если в ходе разработки тщательно фиксируются данные трассируемости требований, у руководителей будет точное представление о состоянии реализации запланированной функциональности и характеристик программного комплекса. Отсутствующие связи от требований указывают на компоненты, которые еще не созданы. Если тестирование дает неожиданный результат, то трассирование связей ***между тестами, требованиями и текстом модулей и компонентов*** могут указать на наиболее вероятные части программного кода, которые необходимо проверить на наличие проявления дефектов. Информация о том, какие тесты проверяют какие требования, экономит время, позволяя удалять лишние, выявлять и создавать необходимые тесты (см. рис. 1.10). Без информации трассируемости возрастает вероятность того, что из тестирования будут упущены компоненты, которые вызовут необходимость добавления, удаления или изменения определенных требований.

Надежная информация трассируемости облегчает соответствующее внесение изменений в ходе сопровождения, что повышает ***производительность разработчиков при модификации комплекса программ***. Информацией трассируемости целесообразно пользоваться при сертификации продукта с особыми требованиями к надежности и безопасности, чтобы продемонстрировать, что все требования были реализованы, хотя это не доказывает, что они реализованы корректно и полностью. Естественно, если требования некорректны или

отсутствуют ключевые требования, то результаты трассируемости не помогут. Документирование взаимосвязей компонентов ***уменьшает риск возникновения проблем***, если вдруг ключевой член команды, обладающей важной информацией о системе, покидает проект. Перечисленные достоинства трассирования требований сложных комплексов программ относятся к долгосрочным выгодам, которые ***снижают общую стоимость жизненного цикла программного комплекса***, хотя при этом увеличиваются затраты на сбор и управление информацией трассируемости.

Отношения трассировки между компонентами проекта могут быть явными или неявными. ***Явная трассировка –*** связь или отношение, между функцией комплекса и компонентом, осуществляющим поддержку этой функции, которая определяется исключительно решением специалиста о том, что такое отношение имеет смысл. Методология разработки и структура системы могут определять ***неявные отношения трассировки – «дочерних»*** требований между компонентами и ***«родительскими»*** требованиями, когда существуют формальные, иерархические отношения. Связи трассируемости помогают отслеживать родительские требования, взаимосвязи и зависимости между отдельными требованиями. Эта информация отражает влияние изменения, если отдельное требование удаляется или модифицируется.

Удобный способ представления связей между требованиями и другими компонентами системы – ***матрица трассируемости требований***. Каждое функциональное требование в такой матрице связано с определенным вариантом использования (в направлении «назад»), и с одним или более элементами верификации и тестирования (в направлении «вперед»). Можно добавить дополнительные столбцы для расширения ссылок на другие рабочие продукты, например, на документацию системы. После того как с помощью инструментального средства заданы все известные отношения между компонентами, обязательным действием является проверка матрицы трассировки взаимосвязей компонентов на наличие следующих ***двух возможных индикаторов дефектов или ошибок***.

Если при просмотре некой ***строки*** матрицы связей не удается обнаружить никаких отношений трассировки, вероятно, что еще не определено требование к программному компоненту, отвечающее функции исходного документа требований. Тем не менее, пустые

строки являются индикаторами возможных ошибок и нуждаются в тщательной проверке. Современные средства управления требованиями должны предоставлять возможность автоматизированного проведения такой проверки.

Если в некотором ***столбце не оказывается*** отмеченных отношений трассировки, вероятно, было создано требование к программному компоненту, для которого нет требующей его функции продукта. Это может указывать на неправильное понимание роли программного требования, недостаток исходного документа проекта, а также на то, что компонент программы неправильный, не соответствует системному требованию или является дефектом разработчика, и в таком случае его следует удалить. Средство разработки комплекса программ должно обеспечивать возможность производить опрос заданных отношений трассировки, а также запоминать эти запросы и вызывать их впоследствии. Трассирование требований нельзя полностью автоматизировать, поскольку многие данные об источниках связей хранятся в головах разработчиков.

Нефункциональные требования, такие, как ***задачи по оценке архитектуры и некоторые атрибуты качества*** не всегда прослеживаются напрямую до компонентов и программного кода модулей. Требование к времени отклика может диктовать выбор определенного оборудования, алгоритмов, структур баз данных или архитектуры комплекса программ. Требования к целостности для аутентификации пользователей активизируют создание производных функциональных требований, которые реализуются, с помощью, паролей. В этих случаях следует трассировать соответствующие функциональные требования в обратном направлении, к их родительским нефункциональным требованиям, и, в прямом, до готового продукта.

Чтобы обнаружить ***пропущенные отношения***, надо искать ***строки матрицы трассировки***, которые показывают, что некая функция ***не*** связана ни с одним программным требованием (прецедентом). При обнаружении пропуска в отношениях нужно вернуться к исходному набору требований к комплексу программ и связанным с ними программным требованиям:

- может оказаться, что связь была случайно пропущена при задании трассировки, в этом случае простое добавление новой связи и пересчет матрицы трассировки решать проблему;
- может выясниться, что при разработке программных требо-

ваний просто не удалось учесть потребности одной из необходимых функций продукта, в этом случае следует исправить проект и добавить подходящие требования.

Обеспечение баланса требований к качеству комплексов программ

Поэтапная разработка спецификаций требований к комплексам программ обычно выполняется итерационно после первичной оценки масштаба проекта и обусловленных такими оценками рисков вследствие ошибок его размера в большую или меньшую сторону. Ограничения при прогнозировании требований определяются, прежде всего, имеющимися данными о прототипах, которые могут быть использованы в качестве исходных аналогов или обобщенных характеристик. В общем случае для оценки, прогнозирования и обоснования ***равновесия требований новой или модификации версии программного комплекса*** необходимы исходные данные:

- обобщенные характеристики использованных ресурсов и технико-экономические показатели завершенных разработок – прототипов программного комплекса, а также оценки влияния на них различных факторов объекта и среды разработки;
- реализованные планы и обобщенные перечни выполненных работ, реальные графики проведенных ранее оценок и разработок различных программных комплексов;
- цели и содержание частных работ в процессе создания предыдущих сложных комплексов программ и требования к их выполнению для обеспечения необходимого качества в целом и минимальных рисков;
- структура и содержание документов, являвшихся результатом выполнения ранее отдельных работ.

Для устранения или снижения рисков до допустимых пределов может быть необходимо изменение требований к функциональной пригодности и/или к характеристикам качества и доступным ресурсам (см. лекцию 1.3). Поэтому на этапах проектирования ***последовательно должны определяться***:

- при проектировании концепции и первичной оценке масштаба проекта – предварительные требования к назначению, функциональной пригодности и к номенклатуре необходимых характеристик качества программного комплекса;

- при предварительном проектировании – уточненная оценка масштаба проекта, требования к функциональным и конструктивным характеристикам качества с учетом общих ограничений ресурсов, перечень источников угроз и величины возможных рисков;

при детальном проектировании – подробные спецификации требований к функциональным и конструктивным характеристикам качества с детальным учетом и распределением реальных ограниченных ресурсов, а также интегральные риски при их оптимизация по критерию качество/затраты.

Этап разработки Концепции проекта целесообразно начинать с формализации и обоснования набора ***исходных требований***, отражающих общие особенности класса, назначения и функции комплекса программ, потребителей и этапов жизненного цикла проекта, каждый из которых влияет на выбор определенных характеристик программного комплекса. Для этого первоначально целесообразно использовать номенклатуру функциональных характеристик и качества, стандартизированных в документе **ISO 9126**. Их описания желательно предварительно ***упорядочивать по приоритетам*** с учетом особенностей назначения и сферы применения конкретного программного комплекса.

На этапе создания концепции и системного анализа формируются цели разработки проекта, выбираются методы и алгоритмы решения основных, функциональных задач, а также формулируются предварительные требования и критерии качества создаваемых компонентов. При этом, естественно, встает вопрос о ресурсах, которые потребуются для достижения целей и о возможности их реализации. Для обеспечения рациональной достоверности первичное прогнозирование целесообразно проводить путем экстраполяции на базе накопленных конкретных данных об отдельных аналогичных предшествующих разработках или с использованием совокупности подобных разработок, проведенных на ***данном предприятии***.

До завершения разработки первичного технического задания на комплекс программ могут быть сформулированы только ***приближенные исходные требования***, отражающие объект разработки и условия его создания. Тем не менее, экспертный опрос ведущих специалистов позволяет составить первичный проект требований, сценарий или модель продукта, масштаб и условий разработки. Достаточно достоверное прогнозирование требований, ре-

сурсов и длительности разработки комплекса программ возможно только при последующем применении жестко регламентированной технологии. После разработки проекта технического задания в процессе структурного проектирования и распределения ресурсов ЭВМ, размер комплекса программ и используемой памяти определяется с точностью 20 – 30%, что позволяет значительно повысить достоверность прогнозирования общих затрат и их составляющих. После выбора технологии, средств автоматизации разработки и технологических ЭВМ появляется возможность более достоверно учесть эти исходные данные для подготовки уточненного сценария разработки. Регистрация этих данных может служить дополнительным ориентиром для оценки полных ***ресурсов на разработку и возможных рисков***.

Разработку и утверждение спецификаций требований к функциональным характеристикам и качеству комплекса программ с учетом анализа рисков целесообразно проводить итерационно на этапах предварительного и детального проектировании. Для критических и особо сложных проектов необходим детальный анализ факторов и рисков при разработке ***сбалансированных требований*** и их оптимизация по критерию качество/затраты. При этом могут применяться два критерия:

- ***максимизация функциональной пригодности*** при применении программного комплекса путем варьирования характеристик качества при заданных ограничениях на использование ресурсов и допустимых рисках;
- ***минимизация риска – ущерба*** от не полного выполнения требуемой функциональной пригодности и некоторых характеристик качества вследствие недостаточных ресурсов и/или низких характеристиках комплекса программ.

Эти критерии целесообразно применять последовательно, итерационно на каждом из основных этапов проектирования комплекса программ. При первоначальном определении требований к функциональной пригодности и к характеристикам качества заданные заказчиком ограничения ресурсов не всегда могут учитывать ряд особенностей проекта, что обусловит недопустимое снижение (или завышение) требований к некоторым характеристикам или рискам. Кроме того, возможно, что некоторые характеристики противоречивы или принципиально нереализуемы в данном проекте. В результате ***не***

сбалансированные требования и доступные ресурсы проявятся как риски – ущерб в виде потерь в качестве или в потребности дополнительных ресурсов. После определения назначения и функций комплекса подготовка исходных данных и концепции проекта должны завершаться ***выделением номенклатуры приоритетных характеристик***, имеющих достаточно сильное влияние на функциональную пригодность и сокращающих риски комплекса программ.

На ***этапе предварительного проектирования*** после фиксирования исходных данных конкретного проекта начинаются выбор требований к его свойствам, а также установление и утверждение конкретных характеристик качества и допустимых рисков, которые могут быть взаимосвязаны. Такой анализ должны проводить заказчик, менеджер проекта и некоторые потенциальные пользователи совместно со специалистами, обеспечивающими ЖЦ комплекса программ и реализацию установленных требований к показателям качества. Для показателей, представляемых качественными свойствами и признаками их наличия, желательно определить и зафиксировать в спецификациях описания допустимых условий и рисков, при которых следует считать, что данная характеристика может или должна быть реализована в проекте. Требования к значениям характеристик должны быть предварительно проверены разработчиками на их реализуемость с учетом доступных ресурсов конкретного проекта и при необходимости откорректированы по составу и значениям с учетом допустимых рисков. При ограниченности ресурсов проекта распределение приоритетов должно становиться более строгим и могут снижаться приоритеты характеристик, для реализации которых ресурсов недостаточно. В результате формируется полный ***набор требуемых функциональных и конструктивных характеристик, свойств и допустимых рисков в жизненном цикле комплекса программ***.

В зависимости от сложности проекта окончательным результатом работ при предварительном или детальном проектировании должны быть ***детализированные и утвержденные требования*** к номенклатуре, свойствам и значениям качества и допустимым рискам проекта, которые достаточны для его полноценного рабочего проектирования и последующей эффективной эксплуатации программного комплекса. Эти ***требования закрепляются в контракте и техническом задании***, по которым разработчик впоследствии должен отчитываться перед заказчиком при завершении проекта. Однако на по-

следующих этапах жизненного цикла и при конфигурационном управлении требования могут изменяться по согласованию между заказчиком и разработчиком, которые чаще всего приурочиваются к подготовке новой версии программного продукта. Для этого необходим мониторинг масштаба проекта, требований и реализаций характеристик и рисков в течение всего жизненного цикла комплекса программ.

При детальном проектировании может быть целесообразно дополнительное уточнение совокупности выбранных функциональных характеристик и качества сложных программных комплексов с учетом рисков, а также соотношения качества и затрат ресурсов. Для обобщенного оценивания качества программного комплекса необходим учет относительного влияния и риска каждой конструктивной характеристики на функциональную пригодность. При этом не всегда учитываются ресурсы для их реализации в конкретном комплексе. Это часто приводит к выдвижению ряда не рациональных требований, которые значительно отличаются либо по степени влияния на функциональную пригодность, либо по величине ресурсов, необходимых для их реализации.

Для целенаправленного эффективного управления рисками комплекса программ при детальном проектировании желательно иметь механизм объединения разнородных характеристик в некоторый интегральный показатель, отражающий их совокупное влияние на его функциональную пригодность. Таким образом, ***при разработке требований*** выявилась проблема анализа системной эффективности программного комплекса и обобщения его характеристик, а также оценивания совместного их влияния на функциональную пригодность с учетом затрат на их реализацию.

Для управления рисками и детального сопоставительного оценивания выбранных характеристик качества целесообразно каждой из них присваивать ***коэффициент или приоритет влияния на функциональную пригодность***. Эти приоритеты могут формироваться формализованным экспертным анализом и детальным установлением влияния каждого атрибута качества на функциональную пригодность с учетом относительных затрат на реализацию соответствующего атрибута. Для этого группа квалифицированных экспертов из состава заказчиков, потенциальных пользователей и/или разработчиков должны оценивать и устанавливать значения таких ***коэффициентов*** –

рисков (приоритетов) в пределах унифицированной шкал (например, 1–10). Аналогично, по такой же шкале экспертами целесообразно оценивать относительные затраты ресурсов, которые следует выделять на реализацию сокращения рисков. Для каждого вида рисков отношение коэффициента влияния на функциональную пригодность к относительным затратам на его достижение можно рассматривать как ***обобщенный уровень приоритета требований к этому сокращению риска***.

Для конкретного комплекса программ состав и значения приоритетов следует ***поэтапно адаптировать и уточнять*** с учетом их назначения и функций. Наивысший приоритет (10) следует интерпретировать как обязательное выполнение разработчиком соответствующего требования к указанному свойству или атрибуту качества с отсутствием риска. Низшее значение приоритета (1) означает, что данный малый риск может не учитываться в данном проекте. Промежуточные значения приоритетов должны отражать относительное влияние соответствующих атрибутов на функциональную пригодность и ее качественные свойства с учетом доступных ресурсов на их реализацию. При этом возможно, что ***некоторые требования*** к атрибутам качества при их низких приоритетах могут не полностью реализоваться в реальном комплексе программ. Это дает возможность разработчикам творчески подходить к требованиям заказчика при реализации жизненного цикла комплекса программ в условиях ограниченных ресурсов. Для последовательного оценивания обобщенных приоритетов при управлении рисками и корректировке атрибутов качества ***конкретного проекта комплекса программ*** целесообразно проводить экспертную оценку коэффициента влияния требуемой характеристики качества на функциональную пригодность и экспертную оценку относительных затрат ресурсов на реализацию требуемых значений качества без рисков.

При проектировании может быть полезно последовательно ***уточнять и выравнивать риски требований*** к каждому свойству и атрибуту качества программного комплекса. Для сложных комплексов программ при уточнении требований к качеству при детальном проектировании целесообразно использовать оценку относительного коэффициента влияния характеристик на функциональную пригодность с учетом затрат на ее реализацию – ***обобщенный уровень приоритета***. При этом набор значений обобщенных уровней приорите-

тов для выбранных атрибутов качества конкретного программного продукта можно разделить, ***например, на три группы***:

- доминирующие характеристики качества и риски, оказывающие наибольшее влияние на функциональную пригодность при допустимых затратах (обобщенный приоритет > 8);
- значения характеристик качества и рисков, имеющие достаточное влияние на функциональную пригодность и значительные затраты на реализацию (обобщенный приоритет < 7, но > 4);
- характеристики качества, значения требований к которым не соответствуют их влиянию на функциональную пригодность и/или затратам на реализацию и могут не учитываться (обобщенный приоритет < 3).

Эти данные могут использоваться, прежде всего, ***как ориентиры*** для селекции и исключения из требований характеристик качества с особенно низкими обобщенными приоритетами рисков, в наименьшей степени влияющих на функциональную пригодность программного продукта и не оправдывающих больших затрат на реализацию. Анализ оставшихся характеристик качества и рисков может проводиться для выделения завышенных требований, а также, возможно, для снижения их значений и приближения их влияния к средним значениям. Кроме того, ***сравнительный анализ баланса обобщенных приоритетов*** на основе отношения влияния на качество и затраты позволяет выделять атрибуты качества, отличающиеся большими затратами – рисками, не оправданными их степенью воздействия на функциональную пригодность. Подобные процедуры могут ***завершать разработку требований к свойствам, характеристикам качества и рискам при детальном проектировании сложных программных комплексов***.

Выше анализировалось и оценивалось преимущественно изменение функциональной пригодности и снижение риска при совершенствовании характеристик качества комплексов программ. Однако для заказчика и пользователей доминирующее значение могут иметь номенклатура и особенности реализации ***некоторых основных функций комплекса программ***, которые, как правило, требуют наибольших затрат и определяют основной эффект и риски от применения программного продукта, а также потенциальный уровень спроса на рынке. Если затраты на разработку можно оценивать и прогнозировать с некоторой достоверностью, то эффективность применения и

особенно будущий спрос на конкретный комплекс программ со стороны пользователей априори оценить трудно. Такие оценки могут проводиться на основе дополнительных маркетинговых исследований и опыта эксплуатации аналогичных комплексов программ или достаточно близких их прототипов.

Часть 2

ТЕСТИРОВАНИЕ МОДУЛЕЙ, КОМПОНЕНТОВ И КОМПЛЕКСОВ ПРОГРАММ

Лекция 2.1

ТЕСТИРОВАНИЕ ПОТОКОВ УПРАВЛЕНИЯ ПРОГРАММНЫХ МОДУЛЕЙ И КОМПОНЕНТОВ

Стратегии выбора тестов для программных модулей

Программные модули являются наиболее массовым компонентом в сложных комплексах программ и ***требуют для тестирования суммарно наибольших затрат***. Затраты на тестирование каждого модуля прямо пропорциональны сложности, которая зависит от его структуры, числа узлов и объема необходимых вычислений. Суммарные затраты на тестирование каждого модуля можно оценить по значению сложности его графа. Некоторые модули на практике могут оказаться недостаточно протестированными из-за их высокой сложности и необходимости больших затрат, которые быстро возрастают (почти квадратично) при увеличении числа узлов и всегда ограничены. Уровень сложности модулей и полнота их тестирования ***определяют качество функционирования групп и комплекса программ в целом***. Наиболее сложные и большого размера модули труднее тестировать и в них наиболее вероятны дефекты и ошибки. Ограниченность ресурсов на тестирование программных модулей приводит к целесообразности по возможности сокращать их сложность, ***выравнивать при проектировании их размеры*** и выделять затраты на их проверку в соответствии со сложностью каждого модуля. Особенно сложные модули полезно делить на более мелкие и простые и так перестраивать их структуру, чтобы сокращалось суммарное число маршрутов в каждом модуле и их размер – рис. 2.1.

Относительная простота программных компонентов и модулей (ПМ) позволяет детально анализировать их внутреннюю структуру и маршруты исполнения программы. Это обеспечивает возможность реализации ***двух стратегий тестирования***: от структуры и от дан-

ных. Этим двум стратегиям соответствуют два метода тестирования программ: метод анализа ***потоков управления*** и анализа ***потоков данных***. Методы дополняют друг друга, и каждый может быть доминирующим на начальных этапах отладки в зависимости от типа объекта и условий тестирования.

Тестирование потоков управления программных модулей и компонентов должно включать:

- стратегии выбора тестов для программных модулей и компонентов:
  - учет особенностей тестирования потоков управления и потоков данных модулей;
  - критерии выделения маршрутов тестирования потоков управления модулей;
  - стратегия упорядочивания маршрутов тестирования потоков управления модулей;
  - планирование тестирования программных модулей;
- анализ сложности тестирования ациклических программных модулей:
  - структурной сложности программных модулей;
  - оценки трудоемкости тестирования программных модулей;
- оценки сложности тестирования программных модулей, содержащих циклы;
- оценки сложности тестирования модулей в программных компонентах;
- оценка корректности результатов тестирования модулей;
- примеры сложности тестирования программных модулей:
  - оценки сложности тестирования элементарных структур модулей;
  - оценки сложности тестирования модулей при различных критериях выделения маршрутов.

Рис. 2.1

Тестирование ***потоков управления*** в структуре модуля программы должно быть начальным этапом, так как при некорректной структуре модуля возможны наиболее грубые искажения выходных результатов и даже отсутствие некоторых из них. Оно состоит в проверке корректности последовательностей передач управления и фор-

мирования маршрутов исполнения модулей комплекса программ при обработке тестов. Для тестирования корректировок структуры модулей программ в большинстве случаев требуются относительно меньшие затраты по сравнению с тестированием потоков данных. При ***первой стратегии*** (рис.2.2) за основу принимается структура ПМ, построенная по тексту программы в виде графа. В графе программы по некоторым критериям выделяются и упорядочиваются маршруты исполнения программы и условия – предикаты, при которых они могут быть реализованы. Эти узлы – условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста. Отклонение процесса исполнения теста от первоначально выбранного маршрута рассматривается как ошибка, причина которой может быть либо в первичной структуре ПМ, либо в реализации конкретного маршрута при заданном тесте на входе.

После устранения ошибок и несоответствия выбранных и реализованных маршрутов ***потока управления*** для каждого из них может проверяться процесс обработки данных и выявляться ошибки в результатах их преобразования. Затем оценивается достаточность выполненного тестирования ***по степени покрытия исходного графа программы*** проверенными маршрутами, которые выделялись по выбранному или заданному критерию. Завершается тестирование при требуемом покрытии графа программы протестированными маршрутами или при использовании ресурсов, выделенных на тестирование. В последнем случае необходима оценка достигнутой корректности программы и регистрация этой величины.

При этой стратегии некоторые выделяемые маршруты могут оказаться принципиально нереализуемыми из-за противоречивых условий в последовательных операторах – узлах графа программы. Вследствие этого для некоторых модулей могут подготавливаться тесты, которые являются избыточными и не отражают реальное функционирование программы. Тем не менее, данная стратегия имеет преимущества при тестировании логических программ с малой долей вычислений. При этом достаточно эффективно контролируется полнота тестирования при различных критериях выделения маршрутов и методах их упорядочения.

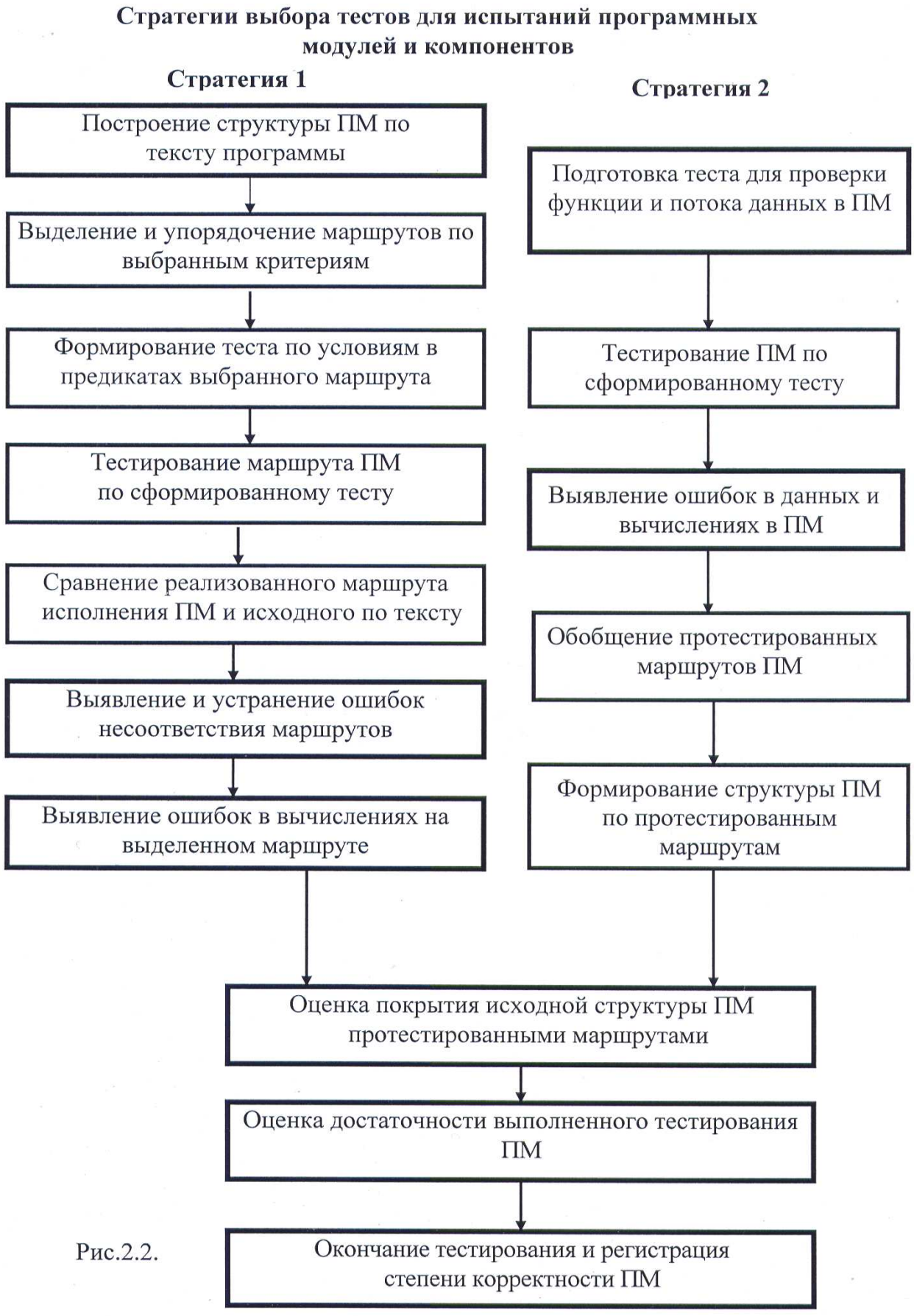

Стратегии выбора тестов для испытаний программных модулей и компонентов
Стратегия 1
Построение структуры ПМ по тексту программы
Выделение и упорядочение маршрутов по выбранным критериям
Формирование теста по условиям в предикатах выбранного маршрута
Тестирование маршрута ПМ по сформированному тесту
Сравнение реализованного маршрута исполнения ПМ и исходного по тексту
Выявление и устранение ошибок несоответствия маршрутов
Выявление ошибок в вычислениях на выделенном маршруте
Стратегия 2
Подготовка теста для проверки функции и потока данных в ПМ
Тестирование ПМ по сформированному тесту
Выявление ошибок в данных и вычислениях в ПМ
Обобщение протестированных маршрутов ПМ
Формирование структуры ПМ по протестированным маршрутам
Оценка покрытия исходной структуры ПМ протестированными маршрутами
Оценка достаточности выполненного тестирования ПМ
Окончание тестирования и регистрация степени корректности ПМ

Рис.2.2.

При ***второй стратегии*** (см. рис. 2.2) за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения расчетов, которые подготавливаются специалистами путем анализа переменных, констант и предикатов в тексте программы. При каждом тесте программа исполняется по определенному маршруту, который регистрируется. При этом реализованный в соответствии с анализируемыми требованиями маршрут и ***поток данных*** рассматриваются как эталонные компоненты структуры программы. По мере тестирования отмечаются проверенные операторы и оценивается полнота ***покрытия тестами требований спецификаций и данных*** на маршрутах тестирования. Накопление и обобщение реализованных маршрутов позволяет воспроизвести структуру программы и контролировать степень покрытия каждого компонента. Для этого на каждом этапе тестирования выделяются компоненты программы, оставшиеся непроверенными, для которых необходимо подготавливать дополнительные тесты. Однако при такой стратегии трудно оценить степень покрытия и проверки всех маршрутов исполнения программы без использования структурного графа, построенного по ее исходному тексту.

Данная стратегия имеет преимущества при сравнительно простой структуре программы и при преобладании в ней вычислительных операторов с множеством переменных и констант. На каждом маршруте упорядоченное варьирование переменных акцентирует внимание на вычислительной части программы, что существенно для такого класса ПМ. Однако возрастает риск пропустить сочетания предикатов, определяющих непроверенный маршрут. Поэтому практически всегда целесообразно ***совместное применение двух стратегий*** с акцентом на одну из них в зависимости от особенностей тестируемого ПМ или его частей. Программы с преобладанием сложной логической структуры и с относительно малой вычислительной частью целесообразно начинать тестировать по первой стратегии и только для маршрутов с вычислительными операторами использовать анализ потоков данных (вторую стратегию). В модулях, имеющих простую структуру и содержащих значительный объем вычислений после первичной отладки по второй стратегии, целесообразна проверка полноты тестирования структуры и завершение отладки по первой стратегии.

Известно, что исчерпывающее тестирование, гарантирующее полное отсутствие дефектов и ошибок, принципиально не возможно

и число дефектов, обнаруживаемых в программах даже независимыми испытателями, имеет пределы. Реальные ограничения доступных ресурсов определяют количество не устраненных дефектов и достижимую корректность компонентов и комплекса в целом. На практике учитывать степень покрытия тестами достаточно трудоемко и зачастую желательно иметь более простой способ регламентирования и оценивания качества тестирования. Возникает проблема, как практически учесть ограничения ресурсов и ***когда прекратить верификацию и тестирование***, а также какая при этом будет достигнута ***корректность программ***, и способна ли она удовлетворить заказчика и пользователя при эксплуатации программного продукта.

В качестве исходной информации ***при тестировании структуры программ*** используется схема связей между операторами текста программы. По этой схеме может выделяться полный набор маршрутов исполнения программы, подлежащих тестированию. Для каждого выделенного маршрута по тексту программы формируется набор условий, определяющих его реализацию и используемый при создании соответствующего теста. Детерминированное тестирование структуры программных модулей имеет целью проверку корректности выделенных маршрутов исполнения программ и обнаружение в основном логических ошибок формирования маршрутов. На практике при отсутствии упорядоченного анализа потоков управления некоторые маршруты в программе (до 50%) оказываются пропущенными при тестировании. Поэтому ***первая задача***, которая решается при тестировании структуры программ, – это получение информации о полной совокупности реальных маршрутов исполнения в каждой программе. Такое представление маршрутов позволяет упорядоченно контролировать достигнутую степень проверки маршрутов и в некоторой степени предохраняет от случайного пропуска отдельных пропущенных маршрутов. В результате значительно повышается достигаемое качество программных компонентов, и тестирование приобретает планируемый систематический характер.

Анализ критериев тестирования и выделение тестируемых маршрутов удобно проводить, используя графовые модели программ. При планировании тестирования структуры программ возникают, прежде всего, ***две задачи***: формирование критериев выделения маршрутов для тестирования ПМ и выбор стратегий упорядочения выделенных маршрутов.

Критерии выделения маршрутов для тестирования соответствуют критериям определения структурной сложности программных модулей. В основном используются критерии:

- покрытия графа программы минимальным количеством маршрутов, охватывающих каждый узел или связь графа хотя бы один раз – f_1;
- выделения линейно-независимых маршрутов на базе понятия цикломатического числа; каждый маршрут отличается от всех других хотя бы одной связью (или одним узлом) – f_2.

Число маршрутов, выделяемых по этим критериям, зависит от структуры программы. В реальных программах часть маршрутов может быть нереализуемой из-за противоречий в условиях, анализируемых на последовательных участках маршрута. Это может приводить к сокращению числа маршрутов по любому критерию. Невыполнение правил структурного построения программ, наоборот, может приводить к возрастанию числа маршрутов. К значительному возрастанию числа маршрутов обычно приводят циклы в программах. Несмотря на перечисленные обстоятельства, приведенные критерии являются достаточно удобными для практических оценок сложности и полноты тестирования структуры программ. Планировать тестирование можно по одному из критериев или используя последовательно более жесткие критерии выделения маршрутов, при которых соответственно возрастает объем и сложность тестирования.

Граф потока управления и/или граф потока данных приведет к различным тестам, и при таком подходе не будет значительного перекрытия созданных тестов. Если поместить граф потока управления внутрь графа потока данных, выявятся повторяющиеся процедуры, большей частью определяемые потоками данных. Если расположите часть, представляемую потоками данных, внутри потока управления, тестовый вариант использует графы потока управления и потока данных.

Стратегии упорядочения маршрутов могут учитывать сложность маршрута и тестов для его проверки (количество узлов в графе, операторов, условных переходов и циклов в маршруте, частость его исполнения при рабочем функционировании ПМ, сложность получения соответствующих эталонных данных и другие факторы). В первую очередь, целесообразно производить проверку основной группы маршрутов с экстремальными значениями используемого критерия в пределах ресурсов, выделенных для тестирования. При имеющихся ограничениях некоторая часть маршрутов может оказаться не прове-

ренной и характеризует достигнутую корректность данной программной компоненты по выбранному критерию.

Упорядочение маршрутов при планировании тестирования в ПМ может базироваться на использовании в основном ***трех характеристик программных модулей (компонентов)***:

- числа узлов (или связей), операторов в выделенных маршрутах или расчетной длительности их реализации (стратегия 1);
- числа узлов – альтернатив или условных переходов, определяющих формирование каждого маршрута (стратегия 2);
- вероятности исполнения реализуемых маршрутов при реальном функционировании программы (стратегия 3).

Эти стратегии тестирования позволяют сосредоточить внимание разработчика на анализе наиболее важных элементов ПМ. При ***стратегии 1*** первичному тестированию подлежат маршруты, наиболее длинные по числу операторов (узлов в графе) и по времени исполнения. Им соответствуют обычно маршруты с наибольшим объемом вычислений и преобразований переменных. Эта стратегия целесообразна при планировании тестирования программ, имеющих вычислительный характер обработки данных при небольшом числе логических условий и маршрутов исполнения программ. Выбранные первичные маршруты не обязательно являются наиболее сложными по логике функционирования.

При ***стратегии 2*** приоритет отдается маршрутам, наиболее сложным по числу анализируемых условий – альтернатив. Такая стратегия предпочтительна при тестировании логических программ с небольшим объемом вычислений. При обеих стратегиях на завершающие этапы тестирования остаются простые по вычислениям или по логике исполнения маршруты, для которых необходимы относительно короткие тесты. Это соответствует традиционной стратегии многих тестировщиков программ подготавливать вначале тесты с возможно большим охватом элементов тестируемой программы.

В типичной программе на долю 3% операторов зачастую приходится до 50% времени исполнения всей программы. В результате изучения профиля программы могут быть выявлены наиболее критические по длительности или самые частые маршруты исполнения ПМ. Использование профиля программ позволяет сконцентрировать усилия разработчиков ПМ на проверке наиболее активно функциони-

рующих элементах и выделить слабо проверенные части программ, которые исполняются редко.

Планирование тестирования структуры программных модулей в значительной степени может быть автоматизировано. Задача автоматизированных систем ***планирования тестирования*** состоит в выделении маршрутов программ по одному или нескольким критериям с последующим упорядочением по заданной стратегии. В результате тестировщик программы автоматизировано информируется о составе маршрутов в программе для проведения ***упорядоченного тестирования***. Кроме того, если фиксировать маршруты, по которым уже проведено тестирование, то их можно автоматически исключать из информирования и выдавать на регистрацию только группу маршрутов, подлежащих первоочередной проверке. Эти же данные могут использоваться для автоматического расчета полноты проверки и для оценки достигнутой корректности программы по каждому из критериев выбора маршрутов.

При анализе сложности тестирования межмодульных связей по управлению и по информации следует учитывать, что тестирование каждой информационной связи обычно необходимо при нескольких значениях каждой переменной, что соответственно увеличивает затраты. Оценка относительной сложности тестирования групп программ способствует правильному распределению ресурсов (машинного времени, труда специалистов и т.д.), выделяемых на тестирование групп компонентов разной сложности. Кроме того, такие оценки стимулируют более полное тестирование ПМ, так как обнаружение и локализация каждой ошибки в группе программ на порядок дороже, чем в автономном модуле.

Сложность тестирования ациклических программных модулей

Сложность программы определяется числом взаимодействующих элементов – узлов, числом связей между узлами и сложностью их взаимодействия. При функционировании программы разнообразие ее поведения и разнообразие входных и результирующих данных в значительной степени определяются ***набором путей – маршрутов***, по которым исполняется программа. Экспериментально установлено, что сложность (трудоемкость) тестирования программного модуля зависит не столько от размера текста программы (числа операторов,

узлов или связей графов), сколько от числа отдельных путей ее исполнения, существующих в программе. Все маршруты исполнения программы модуля и возможной обработки данных должны быть проверены при создании программы и тем самым определяют сложность ее полного тестирования (см. рис. 2.1) .

Выше отмечалось, что маршруты исполнения программного модуля можно разделить на ***два вида***: маршруты принятия логических решений и преобразования логических переменных (потоки управления) и маршруты исполнения преимущественно вычислительной части программы и преобразования переменных (потоки данных). ***Первый вид маршрутов*** является результатом функционирования схем принятия решений и преобразования логических переменных. Для логических переменных отсутствует сильная корреляционная связь между соседними значениями, и каждое изменение переменной может определять разные области пространства результирующих значений на выходе программы. Такое преобразование переменных обеспечивается алгоритмами со сложной логической струкурой, содержащей ряд проверок логических условий, циклов для поиска и селекции переменных, а также логические преобразования переменных. В результате в программе образуется множество узлов ξ_i маршрутов обработки исходных данных, которые определяют сложность структуры программы. В ряде случаев подтверждена достаточно высокая адекватность использования ***потока управления*** и структурной сложности программ для ***оценки трудоемкости тестирования или вероятности*** не выявленных ошибок и затрат на разработку программных модулей в целом. Сложность тестирования потоков управления в программных модулях можно оценивать по числу маршрутов – M_k , необходимых для их проверки, или более по суммарному числу условий – ξ_k (узлов в графе), которые необходимо задать в тестах для прохождения при тестировании всех маршрутов определенной k-й программы:

$$\xi_k = \sum_{i=1}^{M_k} \xi_i \ , \qquad (1)$$

где ξ_i – число условий-предикатов, определяющих i-й маршрут.

Маршруты второго вида обычно логически короче, чем первого, и предназначены для преобразования величин, являющихся зачастую результатами измерения некоторых физических характеристик

(***поток данных***). Такие переменные могут быть связаны условиями гладкости, т.е. условиями малых изменений производных этих переменных по времени или по другим параметрам. При оценке сложности вычислительных маршрутов программ необходим учет числа операндов, участвующих в вычислениях. Кроме того, исходные и результирующие данные при тестировании должны принимать несколько значений. Во всем диапазоне исходных переменных следует выбирать несколько характерных точек (предельные значения и несколько промежуточных), при которых проверяется программа. В особых точках значений и сочетаний переменных и в точках разрыва функции необходимо планировать дополнительные проверки. Таким образом, сложность проверки k-го программного модуля – ξ_k будет определяться числом маршрутов – M_k исполнения программы и числом обрабатываемых операндов – l_i на каждом i маршруте, умноженном на число значений – ν_{ij} для каждой исходной j-й величины на этом маршруте:

$$\xi_k = \sum_{i=1}^{M_k} \xi_I \times \sum_{j=1}^{l_i} \nu_{ij} . \qquad (2)$$

Расчет показателя сложности тестирования программного модуля по такой схеме имеет значительную неопределенность из-за выбора числа значений переменных и констант – ν_{ij} (в основном особых точек и величин) при варьировании исходных и промежуточных значений ***потока данных***. В то же время доля вычислительной части во многих сложных комплексах программ ***управления и обработки информации*** относительно невелика. Ее можно оценить из того, что ***ориентировочное число*** умножений и делений в таких программах в сумме составляет обычно 1 – 3%, а общее число арифметических операций не превышает 10% . Поэтому ниже большее внимание уделено анализу структурного тестирования модулей управляющих комплексов программ.

Структурная сложность программного модуля может быть рассчитана по числу маршрутов в программе – M_k и сложности каждого i-го маршрута – ξ_i . Эти показатели в совокупности определяют минимальную сложность тестов – ξ_k для проверки структуры программного модуля (1), а, следовательно, трудоемкость его тестирования и вероятность пропуска ошибки в программе. Выделение мар-

шрутов исполнения программы, минимально необходимых для ее проверки, и оценка структурной сложности может осуществляться по различным критериям. При этом формирование маршрутов зависит не только от структуры программы, но и от значений переменных, на различных этапах обработки. Затраты на тестирование каждого модуля ориентировочно прямо пропорциональны сложности, которая зависит от его структуры и объема вычислений. При тестировании k-го программного модуля необходимо задать и проанализировать число значений параметров тестов:

$$\xi_k = \sum_{i=1}^{M_k} \sum_{j=1}^{l_i} (\nu_{ij} + \xi_i) \, . \qquad (3)$$

Суммарные затраты на тестирование модуля $\mathbf{C}_k$ пропорциональны значению его сложности и ориентировочно можно определить выражением:

$$\mathbf{C}_k = \mathbf{c} \sum_{i=1}^{M_k} \sum_{j=1}^{l_i} (\nu_{ij} + \xi_i). \qquad (4)$$

Значение множителя трудоемкости тестирования маршрута – **c** зависит от степени автоматизации процесса тестирования и генерации тестов. В высокоавтоматизированных системах тестирования сокращаются затраты ручного труда на подготовку и анализ тестовых данных, однако увеличиваются затраты на машинное время, необходимое для генерации тестов и для автоматической обработки результатов тестирования. В зависимости от этих факторов значения коэффициента – **c** могут различаться в несколько раз, и их следует экспериментально определять для каждой системы автоматизации тестирования.

Для определения суммарных затрат на тестирование модуля необходима оценка вероятности пропуска ошибки в модуле, т.е. достигаемого качества тестирования при принятом критерии выбора маршрутов исполнения программы и числа значений каждой переменной для выбранного маршрута. Обычно устанавливаются некоторые правила выделения маршрутов и варьирования переменных, гарантирующие достаточно глубокое тестирование модулей при доступных

ограниченных ресурсах. Эти правила можно выбирать эмпирически по опыту аналогичных разработок и стандартизировать для определенного предприятия или проекта. Такое выделение маршрутов трудно формализовать, и оно может представляться излишне трудоемким для оценки показателей сложности тестирования программ. Поэтому, зачастую, используются простые критерии выделения маршрутов, учитывающие только потоки управления и структурные характеристики программных модулей, а сложные вычислительные элементы программных модулей выделяются в подграфы и тестируются автономно.

Сложность тестирования модулей, содержащих циклы

Наличие циклов в программе способно резко увеличивать сложность (и трудоемкость) их тестирования. Полное тестирование должно охватывать проверку каждого маршрута в цикле при всех возможных итерациях цикла и при всех сочетаниях циклов с маршрутами ациклической части программы. При возрастании любого из сомножителей (числа независимых ациклических маршрутов, проходящих через цикл, числа внутренних маршрутов тела цикла или числа его итераций) пропорционально растет их произведение, а, следовательно, и сложность тестирования. Поэтому исчерпывающее тестирование реальных сложных программ с циклами практически невозможно (см. рис. 2.1).

В качестве критериев при оценивании сложности тестирования структуры программы с циклами можно использовать два выше подробно описанных критерия: число маршрутов M, выделяемых по критериям f_1 или f_2 ; суммарную сложность тестирования ξ , т.е. суммарное число узлов ветвления на всех выделенных маршрутах, соответствующее числу условий, которое необходимо задать в тестах при выбранном критерии f_1 или f_2. В динамике реального исполнения простейшего цикла между его итерациями могут существовать зависимости, по крайней мере, трех видов:

- на разных итерациях цикла исполняются независимо все возможные маршруты внутреннего тела цикла;
- на всех итерациях цикла исполняется один и тот же маршрут тела цикла или некоторая определенная их последовательность;

- на разных итерациях цикла в силу наличия семантических связей исполняется подмножество реализуемых маршрутов тела цикла, зависящее от данных или от номера итерации.

При ***первой зависимости***, которая встречается наиболее редко, возникает необходимость в полном переборе всех внутренних маршрутов тела цикла в сочетании с определенным числом итераций. В этом случае сложность тестирования цикла определяется сразу обоими параметрами: числом маршрутов тела цикла и числом итераций, и приближается к сложности исчерпывающего тестирования. При ***второй зависимости*** число маршрутов в теле цикла практически не влияет на сложность цикла. Определяющим становится количество итераций, необходимых для тестирования вычислений в теле цикла (например, с учетом требуемой точности). При ***третьей***, наиболее распространенной зависимости, определяющим при оценке сложности тестирования является не число итераций цикла, а число маршрутов тела цикла. Простейшие оценки сложности циклических структур целесообразно проводить в предположении, что порядок реализации маршрутов тела цикла не зависит от номера итерации. В этом случае при оценке сложности циклических структур конструктивным является подход с позиции выбора минимально необходимого числа проверок итераций циклов.

Для оценки ***сложности структурного тестирования*** логических программ с простейшими циклами целесообразно уточнить критерии проверки сложности f_1 и f_2 . По критерию f_1 ациклическая часть программы покрывается минимальным числом маршрутов, в которые входят маршруты, проходящие через цикл, размыкающие его и образующие минимальное покрытие тела цикла. Кроме того, к покрытию добавляется маршрут, содержащий замыкающую дугу цикла.

По критерию f_2 ациклическая часть программы проверяется количеством тестов, равным цикломатической сложности ациклической части программы. При этом к каждому такому маршруту присоединяются все примыкающие к нему циклы. Проверка каждого цикла осуществляется одним маршрутом, содержащим столько итераций, какова цикломатическая сложность тела цикла, а тело цикла покрывается линейно независимыми маршрутами.

При таких оценках определяющими факторами являются полнота проверки тела цикла, условий его замыкания и размыкания. При этих критериях сложность цикла наиболее просто оценить, предста-

вив его эквивалентным ациклическим подграфом (подграфами). Например, при критерии f_2 эквивалентным циклу с одной точкой входа и одной точкой выхода будет линейный подграф, содержащий последовательно соединенные все линейно независимые маршруты тела цикла, связывающие его точку входа с точкой выхода. Такой эквивалентный ациклический подграф добавляется к каждому маршруту в покрытии ациклической части графа по критерию f_2.

Эквивалентной циклу с одной точкой входа и одной точкой выхода при критерии f_1 будет совокупность внутренних маршрутов, соединяющих точки входа и выхода и покрывающих тело цикла по этому критерию, а также маршрут, представляющий собой одну итерацию цикла, состоящую из его замыкающей дуги и маршрута из покрытия тела цикла. Совокупность этих маршрутов, объединенная в ациклический граф с общей точкой входа, будет в этом случае эквивалентным графом для всего цикла. Однако в этом случае к каждому из маршрутов в покрытии ациклической части графа программы по критерию f_1 добавляется только один маршрут из части эквивалентного ациклического графа.

В качестве ***примера предположим***, что число маршрутов в ациклической части программы равно M_1 . Тогда полное множество маршрутов состоит из полной совокупности всех маршрутов M_1 в ациклической части программы и группы маршрутов тела цикла $M_ц$, в которой к каждому маршруту M_1 присоединено 1.2... итерации (витка) цикла, причем на каждой итерации выполняется, по крайней мере, один из внутренних маршрутов $M_ц$ тела цикла. Для графа, имеющего один цикл, требующего исполнения пяти итераций (витков) с тремя внутренними маршрутами, а также содержащего 10 ациклических маршрутов, проходящих через цикл, суммарное число маршрутов для полного тестирования равно $M^* = (3x5)10 = 150$. Такое число тестов трудно реализовать практически и приходится из них выбирать небольшое доступное число тестов, допуская возможные дефекты в не полностью проверенной части программы.

Для ***примера*** выявления типовых структур программ с циклами проанализированы 66 программных модулей, функционирующих в реальных системах управления. Анализ этой выборки показал, что в них отсутствуют циклы с числом выходов больше 3, а число циклов с тремя выходами составляет менее 10% от их общего числа. Число вложенных друг в друга циклов в среднем равно ~ 2,2. Среднее чис-

ло выходов из цикла составляет ~1,4, среднее число операторов ветвления в теле цикла ~ 4. Практически не встречаются сложные зацепляющиеся циклы, т.е. циклы, имеющие общие части, но не вложенные полностью друг в друга. Такие циклы особенно трудно тестировать, в первую очередь, в силу неопределенности значений переменных, влияющих на условия замыкания (размыкания) цикла. ***Применение зацепляющихся циклов*** при создании реальных программ, как правило, ***запрещают*** в методиках проектирования. В реальных программных модулях наиболее часто встречаются простейшие ***одиночные циклы*** с небольшим числом (~ 2 – 5) ветвлений в теле цикла и одним выходом.

Для каждого реализуемого маршрута может быть необходимой проверка при нескольких прохождениях циклов и нескольких значениях каждой обрабатываемой переменной. Особенно важно проверять циклы с условным выходом на одном – двух, промежуточных, а также на максимальном и минимальном витках исполнения циклов. В результате показатель сложности, число необходимых тестов и длительности проверки соответственно возрастают.

Разнообразие реальных циклических структур в программах сильно усложняет их анализ и получение достаточно общих характеристик сложности и достигаемой корректности тестирования. Для обобщенных оценок необходимо в конкретных проектах выделение классов типовых структур сочетания циклических и ациклических частей программ. При применении вложенных циклов со сложной структурой и с большим числом ветвлений в теле цикла сложность тестирования резко возрастает и велика вероятность сохранения в программе не обнаруженных дефектов и ошибок. Поэтому в процессе проектирования программных модулей необходимо в ***максимальной степени упрощать циклические компоненты*** в структуре, ***запрещать*** зацепляющиеся, вложенные и сложные структуры тела цикла, которые во многих случаях при тестировании определяют достигаемую корректность программных модулей.

Выше внимание акцентировалось на сложность тестирования автономных программных модулей. При ***тестировании модулей в составе компонентов*** необходимо учитывать их собственную сложность и межмодульных связей по управлению и по информации. Каждый модуль, протестированный автономно, должен пройти дополнительное тестирование после включения в группу программ и в

составе группы. Затраты на тестирование модулей в составе группы программ должны учитывать относительные суммарные затраты на тестирование влияния взаимодействия в группе всех входящих модулей с коэффициентом d_k <1, зависящим от полноты и качества реализованной автономной проверки k-го модуля. Если модули автономно предварительно не тестировались (например, при нисходящем тестировании), то d_k =1 и затраты на тестирование ***каждого модуля*** войдут в затраты при тестировании группы программ в ***полном объеме***. При тщательном автономном тестировании каждого модуля можно полагать d_k ~ 0,1 – 0,01, т.е. в группе модулей необходимая трудоемкость на его тестирование может составлять только несколько процентов. В результате затраты $\mathbf{C_{групп}}$ на тестирование H модулей с учетом трудоемкости тестирования каждого модуля (4) в составе группы программ могут составлять:

$$\mathbf{C_{групп}} = \sum_{k=1}^{H} d_k \, \mathbf{c} \sum_{i=1}^{M_k} \left(\sum_{j=1}^{l_i} \nu_i + \xi_i \right) \qquad (5)$$

При анализе сложности тестирования межмодульных связей по управлению и по информации следует учитывать, что тестирование каждой информационной связи может быть необходимо при нескольких значениях каждой переменной, что соответственно увеличивает затраты. Оценки относительной сложности тестирования групп программ способствуют правильному распределению ресурсов (времени, труда специалистов и т.д.), выделяемых на тестирование групп разной сложности. Кроме того, такие оценки ***стимулируют более полное автономное тестирование программных модулей***, так как обнаружение и локализация каждой ошибки модуля в составе группы программ на порядок дороже, чем в автономном модуле. Поэтому целесообразно при тестировании групп программ основные ресурсы и время сосредотачивать на обеспечении корректности и высокого качества модулей.

Корректности результатов тестирования графов модулей

Эффективность тестирования обычно определяется полнотой проверки программного модуля или вероятностью наличия не выявленных ошибок в зависимости от затрат на создание тестов, исполне-

ние программ и анализ данных тестирования. Эти затраты в значительной степени зависят от суммарной сложности тестов, проверяющих маршруты исполнения программы. На каждой дуге графа программы между узлами производятся вычисления и преобразования переменных, объем которых может изменяться в широких пределах. Для упрощения анализа тестирования структуры программ предположим, что ***длительность и сложность вычислений на дугах графов программ одинакова и не велика***. Некоторые узлы графа программы могут образовываться в результате схождения дуг без последующего ветвления. Такие узлы не влияют на число маршрутов и их можно при анализе обобщать с ближайшим последующим узлом, в котором происходит ветвление. При этих предположениях сложность теста, проверяющего каждый i-й маршрут, в первом приближении пропорциональна числу дуг графа программы, входящих в этот маршрут, или числу условий ξ_i, которые необходимо задать в тесте.

Полная сложность тестов для проверки программы в последующем принимается равной сумме сложностей тестов ξ_i, используемых для проверки каждого i-го маршрута (1), где суммирование ведется по M_f маршрутам, выделяемым по одному из приведенных выше f -х критериев.

Качество проведенного тестирования и достигнутая корректность модуля определяются возможностью получить при его реальном функционировании искаженные результаты. В маршрутах исполнения программы, содержащих участки, не проверенные тестированием, наиболее возможно искажение результатов из-за не выявленных ошибок. Предположим, что до тестирования вероятность ошибки в j-й дуге графа программы, входящей в i-й маршрут, равна q_{ij}. Кроме того, вероятность ошибки в j-й дуге, в первом приближении не зависит от ошибок в остальных дугах графа программы, т.е. результат вычисления либо полностью используется на этой же дуге, либо является итогом исполнения программы. Тогда вероятность получения правильного результата на конкретном i -м маршруте:

$$\mathbf{p}_i = \prod_{j \in i} (1 - q_{ij}).$$

В реальных условиях конкретное исполнение программы происходит по одному из всех возможных M_f маршрутов. Выбор маршрута определяется обрабатываемыми данными, которые влияют на направление ветвления в узлах графа программы. Реализация каждой j-й ду-

ги i-го маршрута исполнения программы зависит от вероятности π_{ij} выбора этой дуги при предшествующем анализе в узле условия ветвления, вследствие чего вероятность реализации i-го маршрута

$$\pi_i = \mathbf{\Pi}\pi_{ij} \, .$$
$$j \in i$$

В результате вероятность отсутствия проявления ошибки при реальном исполнении программы определяется произведением вероятностей выбора соответствующих дуг и вероятностей правильного исполнения этих дуг:

$$\mathbf{P}_i = \mathbf{\Pi}\pi_{ij} \, (1 - q_{ij}) \, .$$
$$j \in i$$

Предположим, что правильность выполнения конкретного маршрута не зависит от предыдущих исполнений программы и равна $\mathbf{P}_i$. Тогда полная вероятность правильного функционирования программы при произвольных исходных данных определяется вероятностью выбора различных маршрутов и корректностью их исполнения (отсутствия в них ошибок):

$$\mathbf{P} = \mathbf{\Sigma P}_i = \sum_{i=1}^{M_f} \; \mathbf{\Pi}_{j \in i} \pi_{ij}(1 - q_{ij}) \, .$$

Эта величина может рассматриваться как ***показатель корректности программы*** по результатам тестирования ее структуры. Следовательно, вероятность проявления ошибки при случайных данных на входе: $\mathbf{Q} = 1 - \mathbf{P}$.

Значения вероятностей исполнения маршрутов без ошибок зависят не только от вероятностей ветвления в каждой вершине и выбора дуг графа программы, но и от критериев выделения маршрутов. Полное число маршрутов M_f , выделяемых по каждому f-му критерию, может значительно изменяться в зависимости от критерия их выделения. При завершении тестирования программного модуля по всем маршрутам, выделенным по выбранному критерию, модуль следует считать полностью корректным по данному критерию выделения маршрутов. Однако, если после тестирования оценить степень корректности программы при выделении маршрутов по более жесткому критерию, то модуль окажется протестированным не полностью и соответственно вероятность ошибки может быть на равна нулю на

дополнительных маршрутах, выделяемых по этому дополнительному критерию.

Таким образом, набор маршрутов M_f , выделяемых по каждому f-му критерию, является полным в пределах использования этого критерия, и суммарная вероятность исполнения этих маршрутов должна быть равна единице. По каждому набору маршрутов M_f суммарные вероятности реализации маршрутов, подсчитанные по значениям π_{ij} , могут отличаться от единицы. Для оценки вероятности ошибки при выделении маршрутов по каждому критерию значения **P** следует делить на вероятность реализации суммы всех маршрутов по f-му критерию:

$$\mathbf{P} = \sum_{i=1}^{M_f} \prod_{j \in i} \pi_{ij}(1 - q_{ij}) \Big/ \sum_{i=1}^{M_f} \prod_{j \in i} \pi_{ij} \quad .$$

Следует подчеркнуть, что в предыдущем выражении вероятность q_{ij} может стать равной нулю ***после полного*** тестирования соответствующей j-й дуги. Когда тестируется последний маршрут из выделенных по f-му критерию, то предполагается, что входящие в него дуги не будут содержать ошибок после их проверки. Поэтому после тестирования последнего из M_f маршрута вероятность **P** для каждого критерия становится равной единице.

Представляет интерес оценка изменения **P** в зависимости от затрат на тестирование модуля. Они характеризуются суммарным объемом тестов ξ_f для f-го критерия выделения маршрутов $\mathbf{P_f} / \xi_f$, число которых M_f . Для любого числа выделенных M_f в модуле маршрутов эта величина может значительно изменяться в зависимости от критериев их выделения и стратегий упорядочения. При увеличении жесткости критериев выделения маршрутов увеличивается их число и замедляется прирост корректности программ при возрастании суммарной сложности тестов. Так как ресурсы на тестирование всегда ограничены, то приходится прекращать тестирование при использовании допустимых затрат ξ_f . Вспомогательным критерием для оценки целесообразности дополнительных затрат на тестирование может служить величина $\mathbf{P_f}\ \xi_f$. Если она мала, то корректность программ увеличивается медленно и эффективность дополнительного тестирования может оказаться нерентабельной.

Полное тестирование при некотором f-м критерии выделения маршрутов может соответствовать ненулевой вероятности обнаружения ошибки при выделении маршрутов по более жесткому (f+1)-му критерию. Разность этих вероятностей может использоваться для оценки целесообразности перехода на более жесткий критерий выделения маршрутов при ограниченном их числе. Формальный анализ рациональных пределов тестирования и выбора критериев для конкретных программ затруднен разнообразием структур и характеристик программных модулей, а также различием требований, предъявляемых к корректности программ. Поэтому рассмотренные методы следует принимать как ***ориентирующие при планировании и оценке эффективности тестирования структуры модулей программ.***

Вероятность правильного функционирования программы модуля зависит от значений q_{ij} , которые, прежде всего, определяются тем, насколько тщательно протестированы ветви программы к текущему моменту отладки. В не тестированные маршруты могут входить линейные участки программы, ранее тестировавшиеся в составе других маршрутов. Дуги графа программы, проверенные при тестировании некоторых маршрутов, могут содержать ошибки, проявляющиеся при тестировании других маршрутов, включающих те же дуги. О вероятности ошибок q_{ij} в тестированных j-х дугах можно сделать следующие предположения.

Гипотеза 1. При исполнении программы по не тестировавшемуся маршруту возможно получение на дуге ошибочных результатов, несмотря на то, что она уже проверялась в составе другого маршрута. В этом случае вероятность ошибки в любой дуге графа программы не зависит от ее проверок по другим маршрутам и становится равной нулю только после полного завершения тестирования по всем возможным маршрутам, проходящим через данную дугу. Эта гипотеза позволяет получить ***пессимистические оценки*** вероятности правильного исполнения программы после проведения тестирования по части маршрутов. Такие оценки могут оправдываться при наличии сложных вычислений на дугах графа программы. При этом вследствие разнообразия исходных данных, формируемых на других участках маршрутов, проходящих через данную дугу, возможна неполная проверка вычислений до тех пор, пока при различных значениях переменных не будут протестированы все маршруты, включающие рассматриваемую дугу. В результате вероятность обнаружения ошибки при тестировании очередного маршрута зависит от полного числа дуг

в тестируемом маршруте независимо от того, проверялись ли они ранее или нет.

Гипотеза 2. Любая дуга графа программы, проверенная хотя бы в одном маршруте тестирования, проверена полностью и не содержит ошибок, которые могли бы проявиться при других маршрутах исполнения программы. Эта гипотеза соответствует ***оптимистическим оценкам*** вероятности правильного исполнения программы. Подобные оценки близки к реальным значениям для логических программ принятия решений с малым объемом вычислений на дугах графа программы, когда их однократная проверка достаточно достоверна. В этом случае вероятность ошибки при тестировании маршрута зависит от числа только тех дуг в маршруте, которые ранее не были охвачены при тестировании других маршрутов. При всех гипотезах предполагается, что при исправлении новые ошибки не вносятся.

Для получения практических оценок необходимо учитывать диапазон изменения реальной вероятности ошибки в каждой дуге графа программы. Экспериментально установлено, что для слабо структурированных программ число ошибок, выявляемых в процессе тестирования программных модулей, составляет 1–2% от числа строк этих модулей. Для программ обработки информации и управления число условных переходов составляет около 10% от числа строк в программе, т.е. ветвление в программе происходит в среднем после исполнения 10 строк на линейных участках. Следовательно, 10–20% линейных участков (или дуг в графе) программных модулей содержат ошибки перед тестированием, что соответствует вероятности $q_{ij} \sim 0{,}1 – 0{,}2$.

Использование правил структурного программирования спецификаций требований на модули и группы программ позволяет снизить вероятность ошибок приблизительно на порядок, т.е. до уровня $q_{ij} \sim 0{,}01$. Поэтому исследования стратегий тестирования реальных модулей целесообразно проводить в диапазоне $q_{ij} = 0{,}1 – 0{,}01$, соответствующим практически наихудшим и наилучшим реальным значениям.

Примеры оценки сложности тестирования модулей

Для получения ориентировочных характеристик структурной сложности модулей целесообразно оценить характеристики некоторых простейших регулярных структур абстрактных графов программ.

Анализ проведен для двух типов ациклических графов при выделении маршрутов по критерию охвата каждой дуги графа программы хотя бы один раз – f_1, и по критерию выделения всех маршрутов, различающихся хотя бы одной дугой – f_2. В числе выбранных содержались структуры, охватывающие наиболее типовые варианты компонентов графов программ. Структуры различались шириной графов и структурированности ***при одном и том же числе узлов***. Вследствие этого различается число маршрутов и сложность структур. Сложность тестирования модулей оценивалась по числу маршрутов исполнения программы, характеризующих число тестов, необходимых для полной проверки корректности структуры модуля. Кроме того, в качестве показателя структурной сложности модуля анализировалось суммарное число условий, которое необходимо задать в тестах для проверки модуля.

Граф Γ_1 представляет собой симметричное упорядоченное бинарное дерево с максимальным использованием узлов ветвления и максимальной шириной. В графе Γ_2 наоборот - ширина минимальная при последовательных узлах ветвления и всего два полностью различающихся маршрута. Основные результаты анализировались для графов с 30 вершинами, что соответствует средним программным модулям, содержащим около 300 строк (рис. 2.3).

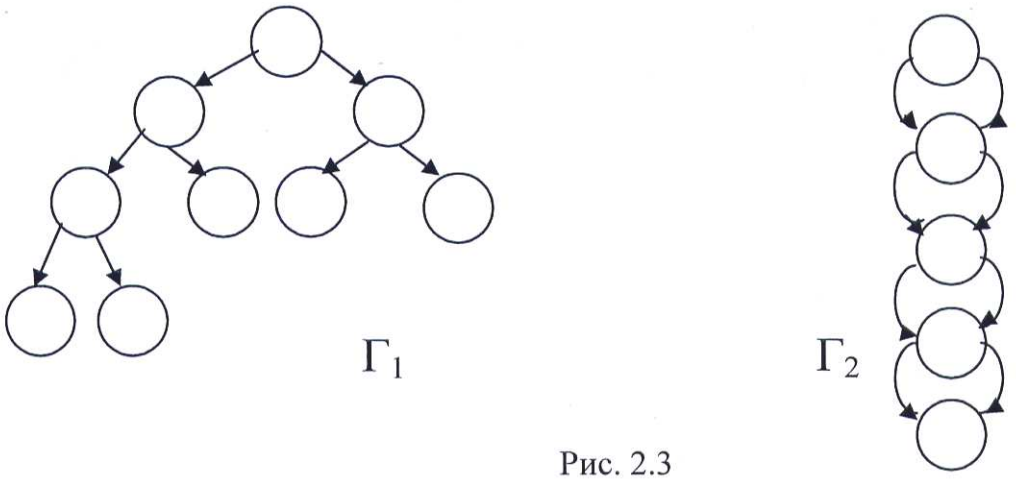


Рис. 2.3

Исследованные графы характеризуются двумя функциональными зависимостями числа маршрутов – M от числа узлов ветвления – $\xi = n_в$. При выделении маршрутов по ***первому критерию*** в графе Γ_2 число маршрутов не зависит от числа узлов и определяется повторяющейся структурой между узлами. В результате в графе Γ_2 выделя-

ется только два независимых маршрута. В графе Γ_1 при выделении маршрутов по первому критерию их число линейно возрастает при увеличении числа узлов.

При выделении маршрутов по ***второму критерию*** их число пропорционально полному числу узлов для всех типов графов. Если в графах учитывать только те узлы, в которых происходит ветвление – ξ , то их число с точностью до единицы равно цикломатическому числу – Z или полному числу маршрутов – M :

$$Z = £ - \xi + 2, \qquad (5)$$

где £ – число связей между узлами.

В графе Γ_2 полное число узлов, кроме вершин с ветвлениями, включает единственную конечную вершину. В бинарном дереве Γ_1 в нижней половине графа узлы не содержат ветвлений и количество таких узлов равно половине от их общего числа. Вследствие этого число маршрутов в таком графе вдвое меньше полного числа узлов.

При ***инженерных оценках числа маршрутов*** в ациклических программных модулях целесообразно пользоваться критерием цикломатического числа и оценивать его по числу ветвлений в графе или условных переходов (альтернатив) в тексте программы. Следует отметить, что в узких графах Γ_2 число маршрутов, выделяемых по первому и второму критериям, различается на величину, пропорциональную числу вершин. Число маршрутов в реальных ациклических программах при втором критерии точно соответствует цикломатическому числу. Таким образом, при инженерных оценках числа тестируемых маршрутов целесообразно использовать выражение (5).

Структурная сложность графов изменяется в более широком диапазоне, чем число маршрутов, что определило выбор логарифмического масштаба графика по оси ординат на рис. 2.4. Наименьшей структурной сложностью характеризуется граф Γ_2 при выделении маршрутов по первому критерию. Для таких графов структурная сложность возрастает практически линейно в зависимости от числа вершин. При выделении маршрутов по второму критерию тот же граф Γ_2 характеризуется наибольшей структурной сложностью. При 32 и более узлов структурная сложность этого графа почти на порядок выше, чем графа Γ_1 при том же числе вершин. Относительная разница сложности этих графов при критерии f_2 приблизительно сохраняется при изменении числа вершин от 16 до 120.

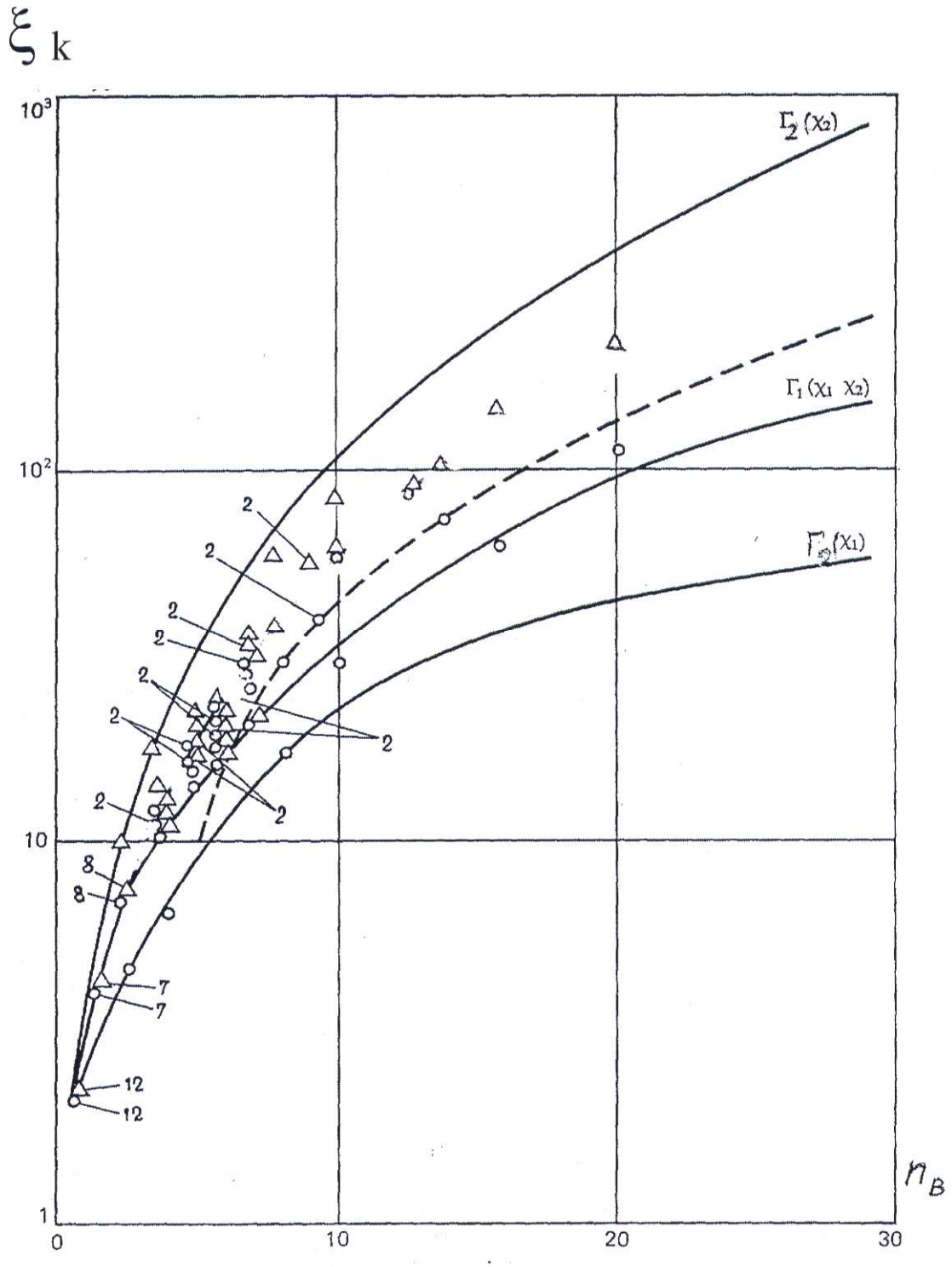


Рис. 2.4

Такое распределение значений структурной сложности от типов графов обусловлено различием их ширины. Так как число маршрутов при критерии f_2 в зависимости от числа узлов $n_в$ во всех графах изменяется одинаково, то определяющим различия структурной сложности является число условий, анализируемых в каждом маршруте. Во

всех маршрутах графа Γ_2 участвуют все его узлы, что определило его наибольшую структурную сложность.

На рис. 2.4 точками (с указанием числа совпадающих значений) приведены значения структурной сложности тестирования ***примеров реальных ациклических программных модулей в двух системах***. Характеристики абстрактных графов Γ_1 и Γ_2 действительно охватывают диапазон изменения показателей сложности тестов для реальных программ, которые по каждому критерию группируются приблизительно посередине. Поэтому при инженерных оценках суммарной сложности тестов, необходимых для полной проверки ациклических программных модулей, можно пользоваться простым произведением числа маршрутов на число узлов в самом длинном маршруте, деленным на два.

Максимальная сложность тестов во второму критерию для произвольных ациклических программ близка к числу $\xi_{max} = n^2$. По тому же критерию минимальная сложность тестов для широких структурированных графов типа дерево (Γ_1) на порядок меньше максимальной. Для усредненных оценок сложности тестов произвольных ациклических программ хорошее ***приближение при*** $\xi > 10$ ***дает выражение*** $\xi = n^2 / 3$ (штриховая линия на рис. 2.4). Характерно, что увеличение числа вершин в 4 раза (от 32 до 128) для рассмотренных графов приводит к возрастанию структурной сложности более чем в 10 раз. Если же программу, имеющую 128 узлов, разделить на 4 модуля, то их суммарная сложность практически равна только учетверенной сложности модулей, содержащих по 32 вершины. Следовательно, для упрощения тестирования программ ***целесообразно ограничивать размеры программных модулей в пределах 10 – 20 узлов*** (около одной тысячи строк). Исследованные реальные программные модули ***систем управления*** в 80% случаев содержат не более 10 узлов и имеют структурную сложность $\xi < 50$.

Примеры оценки сложности тестирования модулей при различных критериях выделения маршрутов. Приведенные выше выражения и критерии можно использовать для априорной оценки числа маршрутов и сложности тестирования графа программных модулей. Для планирования тестирования необходимо выявить зависимости этих характеристик от числа строк в программе модуля ее структуры и критериев выделения маршрутов в графе. Эту задачу можно решать экспериментально (или автоматизировано) путем под-

счета соответствующих характеристик в графах реальных программных модулей, используемых в различных классах комплексов программ. Ниже приводится ***пример такого расчета для среднего по сложности модуля***. Расчеты могут быть автоматизированы, а их результаты обобщены по достаточно большой совокупности модулей комплексов программ определенного класса. Подобные статистические обобщения могут использоваться ***для ориентировки*** при проектировании и сложности тестирования ПМ. Для выявления общих зависимостей сложности тестирования программ целесообразно выделять типовые программные структуры модулей конкретного комплекса, а также такие, которые позволяют получить предельные (сверху и снизу) значения этих характеристик.

На рис. 2.5 представлен ***пример графа модуля программы***, содержащий 14 вершин, 20 дуг и 3 цикла.

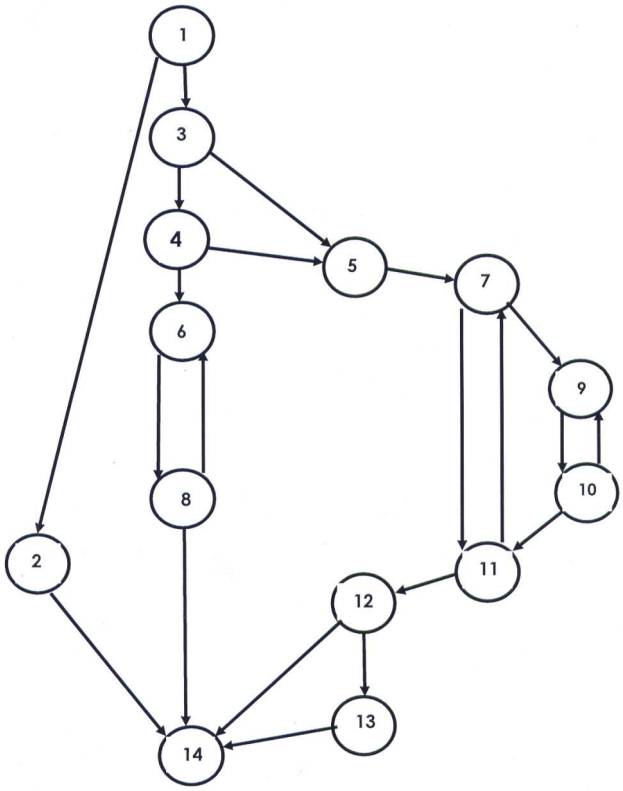


Рис. 2.5

Такая программа сравнительно не высокой сложности содержит 100 – 200 строк ассемблера или около 30 строк на языке программирования высокого уровня и может рассматриваться как достаточно типичная. Для полной проверки модуля ***по первому критерию*** f_1 достаточно четырех маршрутов. По этому критерию гарантируется про-

верка всех передач управления между узлами графа программы и каждой связи узлов не менее одного раза. Самый длинный по количеству узлов последний маршрут не охватывает только 3 вершины из 14 и только 6 дуг из 20. После проверки трех последних маршрутов вне контроля остается один узел и две дуги. Однако при этом критерии не учитывается комбинаторика сочетания условий на разных участках маршрутов, например, при сочетаниях ветвлений в узлах 3 и 12. Сложность программы при выделении маршрутов по этому критерию характеризуется числом маршрутов M = 4 и сложностью тестирования $\xi = 20$. Величина ξ характеризует ***суммарное количество условий***, которое необходимо задать в тестах для полной проверки всех маршрутов, выделенных по первому критерию.

Второй критерий выбора маршрутов f_2 для оценки сложности тестирования структуры использует определение цикломатического числа исходного графа проверяемой программы. Этот критерий наиболее полно исследован при анализе корреляции сложности и трудоемкости создания программ. Данный критерий обеспечивает в исходном графе программы ***однократную проверку*** каждого линейно независимого ациклического маршрута и каждого линейно независимого цикла, в совокупности образующих базовые маршруты. Каждый линейно независимый маршрут или цикл отличается от всех остальных хотя бы одной вершиной или дугой.

Для ***примера графа программы***, представленного на рис. 2.5, множество проверяемых по этому критерию структур образуется из трех линейно независимых циклов и пяти линейно независимых ациклических структур. Эти ациклические структуры исходного графа образуют циклы в максимально сильно связном графе, если конечный оператор искусственно соединить дугой с входным узлом. При этом суммарная сложность тестов, учитывающих все условия прохождения маршрутов ***один раз***, становится $\xi = 26$.

Графы программных модулей может быть удобным представлять в виде ***матриц или таблиц***. Граф программы можно описать ***матрицей смежности*** размера $n \times n$, где n – число узлов – вершин. На рис. 2.6 приведен пример матриц смежности для графа программы, представленного на рис. 2.5.

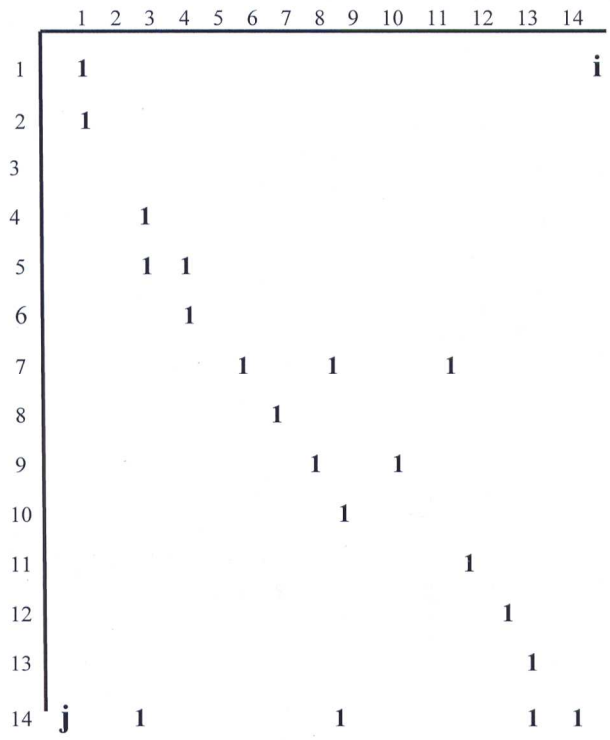


Рис. 2.6

В этой матрице единица располагается в позиции [i, j], если из вершины i можно перейти к вершине j за один шаг. В противном случае в позиции матрицы размещается 0 (на рис. 2.6 – опускается). Маршруты в программе можно представить ***матрицей достижимости***, в которой на позиции [i, j] помещается 1, если из узла i можно перейти в узел j ***за любое число шагов***. Описание структуры программ матрицами смежности и достижимости позволяет удобнее вести численный анализ графов и автоматически рассчитывать цикломатическую сложность программных модулей.

Позиции в реальных матрицах оказываются по большей части пустыми (см. рис. 2.6). Это бывает часто, поскольку в графах число узлов, как правило, не превышает значительно число связей. Несмотря на то, что матрицы компактны и их проще рисовать, чем графическую форму модуля, использование такого формата часто затруднительно и приводит к ошибкам при работе с большими матрицами. Если связи обладают весами, они должны в матрице дополняться соответствующими значениями веса.

Анализ матриц достижимости позволяет сравнительно просто выделять циклы и некоторые ***структурные некорректности*** (лишние и тупиковые участки графа программы). Номера строк, в которые входят циклы, можно выделить, отмечая диагональные элементы,

равные 1 и одинаковые для нескольких строк элементы, составляющие строку. На рис. 2.6 одинаковыми являются 6 и 8 строки, которые имеют на диагонали 1, в результате чего образуется маршрут с циклом между 6 и 8 вершинами. Аналогично можно выделить еще два маршрута с циклами в 13 строке и в 7, 9, 10, 11, 12 одинаковых строках.

Проектирование тестирования потоков управления модулей

Представленные в данной лекции метод стратегии оценки и использования графов потоков управления в программных модулях целесообразно применять при формировании и автоматизации технологии, методики и средств тестирования модулей. Основой могут служить традиционные процессы тестирования: анализ объекта и оценка сложности тестирования, подготовка и исполнения тестов, анализ и корректность результатов тестирования, выявления и ликвидации дефектов и ошибок. Базой значительной части этих процессов могут быть символические модели программ в виде графов, потоков управления и маршрутов исполнения программных модулей, создаваемые на основе преобразования текстов программ на языках программирования. Такая обобщенная визуализация текстов и структуры программных модулей позволяет наглядно представлять логические процессы реализации сложных функций программ, оценивать необходимые ресурсы для тестирования и сопоставлять их с требованиями и эталонами. ***Проектирование тестов для модулей на базе потоков управления, их выполнения и устранения ошибок*** целесообразно использовать при разработке технологий и Программ тестирования программных модулей и компонентов для обеспечения их соответствия требованиям высокого качества по следующим этапам (рис. 2.7):

- подготовка и построение модели графа узлов и связей модуля на основе текста программы;
- выделение, анализ и обеспечение корректности циклов, графа модуля программы;
- выделение и тестирование ациклических маршрутов графа модуля программы в соответствие с требованиями к качеству;
- фиксирование результатов исполнения каждого теста на выбранном маршруте, устранение дефектов и ошибок и завершение тестирования модуля.

Проектирование тестов для потоков управления модулей, их выполнение и устранение ошибок должно включать:

- подготовку и построение модели графа узлов и связей программы модуля:
  - трансляцию текста программы модуля, полученного от программиста в графовую модель узлов и связей между ними;
  - однозначную нумерацию предложений текста программ модуля и преобразование в имена узлов графа;
  - преобразование текста программы модуля в виде коротких предложений и предикатов на эквивалентные последовательности узлов и связей графа модуля;
- выделение, анализ и обеспечение корректности циклов графа программы:
  - выделение в графе модуля циклов, расчет их сложности и развертывание для автономного анализа и тестирования итераций циклической части маршрутов;
  - тестирование выделенных циклов по выбранным внутренним маршрутам и итерациям;
- выделение и тестирование ациклических маршрутов графа модуля программы в соответствие с требованиями к качеству;
  - расчет сложности и проверка первичной корректности структуры ациклического графа модуля для исключения структурных ошибок программистов;
  - анализ взаимной корреляции предикатов в графе, выделение и исключение нереализуемых путей вследствие противоречий между предикатами;
  - выделение и упорядоченье маршрутов тестирования по сложности, длительности или вероятности исполнения;
  - тестирование выбранных маршрутов, при входных величинах, соответствующих проходу по выбранному пути:
  - согласование с программистами промежуточных численных расчетов и предикатов, которые необходимы для проверки корректности реализации маршрутов;
  - активизирование выбранных маршрутов тестирования по выбранному пути;
- фиксирование результатов исполнения каждого теста на выбранном маршруте, устранение дефектов и ошибок модуля:
  - определение соответствия тестирования модуля предполагаемому результату, оценка качества, обнаружение и исправление ошибок;
  - подтверждение и документирование полной корректности выбранных маршрутов потоков управления модулей и завершение тестирования.

Рис. 2.7

Лекция 2.2

ТЕСТИРОВАНИЕ ПОТОКОВ ДАННЫХ ПРОГРАММНЫХ МОДУЛЕЙ

Свойства и тестирование потоков данных программных модулей

В ***графах потока данных*** значения объектов связаны с обрабатывающими узлами, в которых эти значения вычисляются. При применении графов потока данных узел может быть использован для представления нескольких различных вычислений и, таким образом, может иметь несколько различных объектов, ассоциированных с исходящими связями. Обычная практика – размещение имен объектов на связях, исходящих из таких узлов. При таком использовании эти имена характеризуются значениями исходящих связей. Поскольку исходящие из одного узла связи являются входящими по отношению к следующему узлу, они называются значениями входящих связей. Узел выбора данных вычисляет специальную функцию значения входящей связи для использования в исходящей связи. ***Входящие связи*** узла выбора данных должны быть помечены условием предиката, при котором выбирается данная входящая связь. Его значение выбирает объект данных, соответствующий входящей связи. Обрабатывающий узел с одной или более входящими связями формируется, по крайней мере, с одной исходящей связью. Входящие связи обозначаются именами данных. ***Исходящая связь*** обозначает вычисленную функцию этих объектов данных. Запоминающие узлы при последующей обработке должны уточняться, какое именно значение переменной используется: значение, сохраненное на диске, или значение в ОЗУ, предшествующее сохранению. При использовании в сетях с файлами совместного доступа они могут отличаться (и таким образом быть источником ошибок).

Тестирование ***потоков данных*** можно разделить на два этапа. ***Первый этап*** тестирования состоит в анализе узлов обработки данных, определяющих значения предикатов в операторах выработки логических решений при взаимодействии элементов в модуле программ. Эти решения влияют на изменения маршрутов обработки ин-

формации, что сближает в этой части метод тестирования потоков данных с тестированием структуры модуля. ***Второй этап*** – тестирование обработки данных на соответствие заданным требованиям. Оно состоит в проверке вычислений по аналитическим формулам или определение численных значений результатов решения задач в зависимости от числовых или логических значений исходных данных.

Функционирование программного модуля можно рассматривать как ***обработку потока данных*** (переменных, предикатов и констант), передаваемых от входа в программу к ее выходу. Входные данные последовательно используются для определения ряда промежуточных результатов вплоть до получения набора выходных данных. Задача тестирования потока данных состоит в установлении корректности их обработки и в выявлении ошибок в тестируемой программе. Эта задача может решаться статически без исполнения программы (анализом ее текста и графа) и динамически – путем исполнения программы на ЭВМ в машинных кодах при различных исходных данных.

Методы тестирования потоков данных более эффективны, чем методы тестирования потоков управления, в том смысле, что они могут обнаруживать больше типов ошибок. Они также более эффективны с точки зрения теории, поскольку можно создать все тесты с помощью тестирования потоков управления и еще некоторые дополнительные тесты для потоков данных. Однако за это приходится расплачиваться. Чтобы достичь более эффективного тестирования, приходится проделать больше работы. Больше работы, во-первых, при проектировании тестов, а во-вторых, при проверке результатов тестирования. Приведенные ниже методы тестирования, используемые в графах потоков данных, расположены по степени возрастания их эффективности.

Модель потока данных – это ***способ структурировать мышление***, необходимый для получения полезных тестов. Если сделать отдельно модель потока данных и модель потока управления, каждая приведет к различным тестам, и при таком подходе не будет значительного перекрытия созданных тестов. Поэтому полезно помещать модель потока управления ***внутрь модели потока данных***. Повторяющиеся процедуры в модели большей частью определяются потоками данных. Можно поместить модель потока данных ***внутрь модели потока управления***. Тогда обработка моделируется при помощи графа потока

данных. Выбор способа совмещения моделей потоков зависит от того, какой из них является более сложным и доминирующим.

Данные, участвующие в вычислениях, должны быть определены явно в потоках данных по имени, типу и способу доступа. Это позволяет рассматривать программу в виде мультиграфа, заданного структурой передач управления ***(потоком управления)*** и графом преобразования данных, участвующих в вычислениях ***(поток данных)***. Пересечение потоков управления и потоков данных осуществляется в узлах ветвления: проверки условий и циклов. Совместный анализ потоков управления и данных позволяет проверять корректность областей определения переменных на маршрутах исполнения программы. Последствия ошибок данных в модуле могут проявляться как малые изменения некоторых переменных в процессе вычислений и как полное искажение или отсутствие на выходе требующихся величин. Тестирование программного модуля целесообразно проводить на упорядоченных наборах данных с учетом степени их влияния на выходные результаты. С этой позиции для последующего анализа целесообразно выделить ***два вида обработки данных***: способный полностью изменять область определения результатов и изменяющий результаты в пределах некоторой ограниченной области определения.

Первому виду обработки соответствуют исходные данные в критических точках (узлах) и на границах областей изменения переменных. При таких критических значениях может ***изменяться маршрут*** исполнения программы, вследствие чего возможно наибольшее изменение результатов. Поэтому обычно тестирование обработки данных, прежде всего, направлено на проверку исполнения программ при значениях переменных, влияющих на выбор маршрута и логику функционирования программ (стратегия областей).

Граничные условия – это ситуации, возникающие в непосредственной близости к границам областей изменения обрабатываемых переменных. Число таких критических значений каждой переменной может быть на несколько порядков меньше, чем число значений по всей внутренней части области изменений этой величины. Большинство критических значений (предикатов) может существенно влиять на результаты и подлежит наиболее тщательному тестированию. В этой части тестирование обработки данных по содержанию близко к тестированию структуры (потока управления) программы (см. лекцию 2.1).

При этом виде тестирования маршруты формируются в процессе анализа и обработки данных на последовательных операторах условий в тексте программы. Таким образом, все множество маршрутов является реализуемым и определяется составом реальных тестовых данных. Сочетания исходных данных в тестах непосредственно влияют на степень охвата тестированием участков программы модуля. Путем сопоставления проверенных маршрутов с маршрутами, выделенными по графу программы при различных критериях можно оценивать полноту тестирования потока управления модуля и приблизительную его корректность.

Второму виду обработки соответствуют данные в ограниченной или неограниченной области определения, которая может делиться на некоторое множество сопрягающихся областей (подобластей). Изменение данных в пределах такой области не влияет на маршрут исполнения программы. Поэтому для проверки функционирования программы из всего множества значений достаточно использовать при тестировании только несколько значений внутри и вблизи границ области. Количество величин, используемых для тестирования при обработке этого вида, может быть на несколько порядков меньше полного числа значений каждой переменной в области. В процессе такого тестирования проверяется точность осуществляемых вычислений, правильность размерности обрабатываемых величин, корректность формирования логических величин и т.д. При этом тестирование должно охватывать всю область изменения каждой обрабатываемой переменной и каждой результирующей величины.

Проектирование и выполнение тестов в модели потока данных остается почти таким же, как в модели потока управления (см. рис. 2.7). Процессы формирования и тестирования модели потоков данных (без циклов) представлены на рис. 2.8. ***Тестирование с применением графовых моделей*** программных модулей способно обеспечивать их высокое и управляемое качество. Оно состоит из двух технологических этапов: построения и анализа графовой модели для подготовки тестирования и этапа планирования, реализации и исполнения тестов для выявления дефектов и ошибок в программе. При планировании тестирования на основе объединенной графовой модели потоков управления и данных подготавливаются исходные данные требований и эталонов и обобщенный граф модели модуля. Эти данные используются для определения предикатов и переменных правильных маршрутов исполнения программы и описаний тестов.

Формирование и тестирование потоков данных модуля должно включать:

- построение и анализ графовой модели для подготовки тестирования:
  - идентификацию входных переменных и констант, присвоение каждой входной переменной имени и создание для нее входного узла;
  - преобразование спецификации и текста программы модуля, чтобы каждой вычисляемой функции соответствовало одно предложение – подграф;
  - перечисление функций – подграфов и создание структуры узлов модуля, пока не охвачены все потоками данных;
  - формирование списка, в котором находятся промежуточные подграфы, зависящие от большого количества вычислений;
  - проверка промежуточных функций, можно ли упростить модель, удаляя промежуточные узлы между подграфами;
  - проверка возможности упростить модель, добавляя промежуточные узлы и подграфы для сложных вычислений;
  - регистрация модели потока данных модуля – функций, связей и подграфов, которые отражают их обработку;
  - проверка полноты и корректности модели потока данных;
- планирования, реализации и исполнения тестов для выявления дефектов и ошибок в программе:
  - выбор маршрутов тестирования с использованием порожденных подграфов;
  - предсказание и регистрация ожидаемых результатов – эталонов тестирования выбранных маршрутов и подграфов;
  - определение критериев соответствия эталону для каждого теста;
  - выполнение тестов потоков данных на выбранных маршрутах и подграфах;
  - сравнение результатов тестирования маршрутов и подграфов с эталонами, регистрация дефектов и ошибок;
  - исправление дефектов и ошибок и регрессионное тестирование маршрутов и подграфов.

Рис. 2.8

Маршруты формируются в соответствии с заданной стратегией и критерием выделения и упорядочивания маршрутов. Сформиро-

ванные тесты используются для контроля исполнения программы компонента и выявления ошибок.

Подготовка тестов потоков данных модуля начинается с проверки корректности его спецификации, идентификации входных переменных и констант, присвоение каждой входной переменной имени и создание для нее входного узла. Текст программы модуля преобразуется так, чтобы каждой вычисляемой функции соответствовало одно предложение. Создается структура компонента, начиная с элементов, которые зависят от входных переменных и от результатов предыдущих функций, пока не охвачены все. Производится проверка промежуточных функций, является ли их упорядочение необходимым или просто удобным, можно ли упростить модель, удаляя промежуточные узлы или, добавляя промежуточные узлы и подграфы для сложных вычислений. Следует ***зарегистрировать модель потока данных*** – поименованный набор узлов – функций, их связей и подграфов, которые отражают их обработку.

Планирование и реализация тестирования модуля с использованием графов потоковых моделей программ начинается с формирования задания, эталонов и ограничений для планирования тестирования программного модуля (см. рис. 2.8). На основе подготовленного графа программного компонента должно производиться выделение ***циклических подграфов*** для автономного тестирования, использование предикатов и маршрутов циклов для формирования набора тестов и выполнение их тестирования. После этого целесообразно осуществлять выбор маршрутов тестирования по узлам и порожденным ими сложным вычислительным подграфам и активизирование тестирования порожденных подграфов. В результате возможно преобразование объединенной графовой модели к ациклическому виду. В этой модели производится выделение маршрутов по выбранному критерию и упорядочение выделенных маршрутов графа по выбранной стратегии ациклического графа исполнения программы. Далее целесообразно провести анализ и исключение нереализуемых маршрутов графа по противоречивым условиям в предикатах или на граничных значениях переменных. Оставшийся список предикатов и выделенных маршрутов используется для предсказания и записи ожидаемых итогов тестирования маршрутов, формирования набора тестов и реализации тестирования по выбранным маршрутам и подграфам. Это позволяет оценить полное число тестов, использованных для тестирования выделенных маршрутов с учетом значений эталонов и ог-

раниченных ресурсов, а также степень покрытия модуля тестами, проконтролировать результаты тестирования, выделенные дефекты и ошибки. При этом должна быть проведена оценка качества программного модуля и допустимости его применения в проектируемом комплексе, подтверждена корректность значений результатов в промежуточных узлах, циклах и подграфах. Все итоги тестирования должны быть документированы. Ниже представлены некоторые методы тестирования, используемые в графах потоков данных при выборе маршрутов в порядке возрастания их эффективности.

Покрытие ввода/вывода. Для каждого выходного узла используется набор входных значений, которые приводят к определенному значению вывода. В этом методе есть слабое место: покрытие входных и выходных узлов и, возможно, некоторых промежуточных узлов, но если среди них имеется отдельный предикат выбора, можно быть уверенным, что используется только одно значение этого предиката, другие не будут протестированы. Эта проверка слишком слаба, чтобы быть полезной. Она гарантирует только то, что программный модуль работает для одного набора входных данных.

Ввод/вывод + все предикаты. Усиливается покрытие ввода/вывода так, чтобы все предикаты (включая предикаты потока управления для циклов и другого необходимого упорядочения) были проверены для обоих значений истинности и аналогично для предикатов в операторах. Это лучше, но если есть промежуточные вычисления, результаты которых не используются, они не обнаруживаются. Вычисление чего-либо без последующего использования полученного результата – это, скорее всего, ошибка, которая чаще всего является лишь неоправданной тратой ресурсов, но иногда может быть опасна. Следовательно, промежуточное вычисление, выход которого не используется, по всей видимости, является ошибкой, и ее следует обнаружить. Если просто проверены все узлы с предикатами (как потока управления, так и выбора данных), это больше, чем тестирование ветвей потока управления (из-за проблемы составных предикатов), но не получены все преимущества, которые дает тестирование потоков данных.

Частичное покрытие узлов. Предыдущие методы обеспечивали покрытие некоторых узлов и некоторых связей, но при этом не гарантировали, что будет обеспечено покрытие всех узлов и/или всех связей. Это называется стратегией ***всех определений***, поскольку каждый вычислительный узел в модели потока данных соответствует опреде-

лению некоторой переменной. Однако это не обеспечивает полное покрытие связей. Таким образом, эту стратегию нельзя прямо сравнивать со стратегией ***ввод/вывод + предикат***, описанной ранее. Это означает, что каждая функция была выполнена хотя бы один раз и дала правильное значение для этого случая. Проведена проверка промежуточных результатов, поскольку тестирование каждого вычислительного узла и подграфа подразумевает проверку вычисления, сделанного в этом узле.

Все узлы покрывают узлы, а не только вычислительные – обеспечено покрытие всех узлов выбора данных и узлов потока управления. Но это недостаточно сильно, поскольку не дает гарантии, что проверены все варианты для предикатов выбора и потока управления.

Покрытие связей – охват всех связей в графе потока данных. Она соответствует стратегии ***всех использований вычислений***. Всякий раз, когда вычисление выполнено, будет проверяться каждое использование результата этого вычисления подграфа в последующей обработке. Это подразумевает проверку всех промежуточных вычислений, а не только конечных выводов. Но это не означает тестирования каждого возможного пути в программе. Это даже не тестирование всех возможных путей между точкой определения и местом его последующего использования.

Тестирование графов модулей программ с учетом значений переменных и констант

Анализ и учет при тестировании областей определения значений переменных и констант. При анализе обработки данных в пределах внутренних областей их определения методы тестирования целесообразно применять упорядоченно в ***следующей последовательности***:

- тестирование корректности записи и считывания переменных при вычислениях и полноты состава выходных данных на всех маршрутах исполнения программы;
- тестирование точности результатов вычислений и корректности обработки каждой переменной или константы;
- тестирование на полное соответствие состава и значений выходных данных требованиям программной спецификации.

В приведенной последовательности частные методы тестирования обработки данных позволяют, прежде всего, выявлять первичные

ошибки, которые способны искажать результаты в наибольшей степени. При ограниченных ресурсах и такой последовательности тестирования в программе будут оставаться ошибки, наименее влияющие на корректность выходных данных. Полезно акцентировать внимание на выявление ошибок обработки данных, которые влияют на логику исполнения программы, запись и считывание переменных, полноту состава результатов, точность расчета выходных данных и полное соответствие выходных данных требованиям спецификации модуля или компонента. На основе таких проверок может оцениваться степень охвата тестированием всех условий, определенных в спецификации, и дополнительное тестирование следует проводить только при отдельных недостаточно проверенных входных или результирующих данных – рис. 2.9.

Тестирование графов модулей программ с учетом значений переменных и констант должно включать:

- анализ и учет при тестировании областей определения значений переменных и констант;
- учет граничных значений переменных и констант при тестировании потоков данных;
- тестирование корректности определения и использования данных на маршрутах исполнения программ;
- тестирование корректности и точности вычислений каждой переменной;
- определение и регистрация не реализуемых маршрутов потоков данных программных модулей.

Рис. 2.9

Маршруты последовательности обработки данных могут зависеть от любых типов величин. При выборе направления ветвления в узлах графа программы участвуют переменные и константы, отражающие вещественные, целые, булевские, символьные, векторные и другие величины. Области определения таких величин зависят от их типов и содержания и могут представлять как отдельные точки и несвязанные области, так и неограниченную непрерывную последовательность значений. Одной из задач тестирования является проверка сопоставимости сравниваемых типов величин и идентичности условий их кодирования (разрядности, масштабов и т.д.). Критические

значения – предикаты, влияющие на маршруты, во многих случаях не являются фиксированными, а формируются при сопоставлении нескольких переменных или при их преобразованиях. При этом предикаты образуются во всей области изменения каждой из переменных, например, когда они оказываются равными или отличаются на некоторую постоянную величину.

Предикаты, определяющие выбор маршрутов исполнения программных ***модулей систем управления***, формируются в результате вычислений в узлах обработки на линейных участках программы. Эти участки в среднем невелики и содержат около 10 операторов программы. Вычисления в большинстве случаев представляют собой простейшие линейные преобразования входных или промежуточных данных. По оценкам 95 – 97% арифметических операторов включают только сложение и вычитание, а 98% всех выражений между последовательными предикатами содержат не более двух операторов. Кроме того, предикаты обычно очень простые – в большинстве случаев с одной входной переменной и практически отсутствуют предикаты, использующие более двух входных переменных. Нелинейные предикаты встречаются очень редко (0,1 – 0,3%). Приведенные данные позволяют ограничить анализ линейными предикатами, характерными для широкого класса программ управления.

Учет граничных значений переменных и констант при тестировании потоков данных. Каждая ограниченная область исходных данных соответствует определенному маршруту в программе. ***Граница области*** определяется интерпретациями предикатов по маршруту и состоит из набора участков границы, каждый из которых определяется единственным простым предикатом, формирующим дугу маршрута в графе программы. Каждый участок границы области может быть открытым или закрытым в зависимости от оператора условий в предикате (рис. 2.10). Закрытый участок границы принадлежит ограничиваемой области и формируется предикатами с операторами $\leq$, $\geq$ или $=$. Открытый участок границы не входит в состав области и формируется операторами $<$, $>$ и $\neq$. Общее число предикатов в маршруте – это верхний предел числа граничных участков области входных переменных данного маршрута, так как некоторые предикаты маршрута могут в действительности не создавать граничных участков. Такие случаи возникают, когда предикат требуется для нескольких путей и в некоторых из них повторно анализируется на маршруте.

Предикаты можно разделить на три типа: равенство (=), отношение больше – меньше ($<$, $>$, $\leq$, $\geq$) и неравенство ($\neq$). Использование предикатов каждого типа дает существенно различный эффект на границе области. Предикаты каждого маршрута определяют некоторую гиперплоскость в пространстве. Ограничения типа неравенства эквивалентно составному предикату (A<B) и (A>B), такой предикат приводит к разделению исходной области на две части. Предикаты типа равенства и больше – меньше приводят к формированию области в виде единственного многоугольника. Многоугольник является выпуклым, если для любых двух точек области участок границы, формирующий маршрут, входит в состав этой области. Если в состав предикатов ввести неравенства, то пространство входных данных будет объединением множества выпуклых многоугольников.

Таким образом, программа по отношению к потоку данных, прежде всего, выполняет функцию ***разделения пространства исходных данных на области***, каждая из которых соответствует одному исполняемому маршруту. Ошибки в программе могут быть обусловлены модификацией границы области определенного маршрута, приводящей к расширению или сужению пространства исходных данных соответствующего маршрута. Кроме того, деформация границ областей может приводить к ошибкам уничтожения некоторых областей и потере соответствующих им маршрутов. Причинами таких ошибок могут быть искажения операторов анализа условий или искажения в процессе вычисления значений предикатов при правильном содержании оператора условия. В последнем случае обычно сдвигается граница области, однако она сохраняет общую структуру. Искажения операторов анализа условий может приводить как к деформации границы области, так и к появлению новых границ или их уничтожению, вследствие чего области могут разделяться или сливаться.

Один из достаточно часто встречающихся типов ошибок обусловлен искажениями условий формирования границ областей. Выбор тестовых данных вблизи границ областей обеспечивает наибольшую чувствительность к этим ошибкам. Тестовые исходные данные в зависимости от их положения относительно конкретной границы области можно разделить на два вида. Первый вид данных (***принадлежащая*** точка) размещается на границе области. При тестировании граничные точки входят в состав проверяемой области (условия $\leq$, $\geq$

, =). Второй вид (***непринадлежащая*** точка) отстоит от границы на сколь угодно малую величину и находится на открытой стороне данной области (условия <,>, ≠). При тестировании такой области принадлежащие (граничные) точки относятся к смежной области, а в проверяемую область данная граница не входит.

Для проверки границ областей разработана стратегия выбора тестовых значений, минимизирующая объем проверок. Для закрытой области на каждой границе целесообразно формировать три тестовых значения. Первые два теста (A и B) (см. рис. 2.10) размещаются на границе данной области вблизи стыка данной границы с соседними. Третья точка (C) размещается на малом расстоянии □ от данной границы и удовлетворяет всем неравенствам области, кроме проверяемого на данной границе. При таких тестовых данных должны получаться выходные результаты, искажения которых обусловлены невыполнением условия на анализируемой границе.

При любом сдвиге границы областей на величину, большую □, обнаруживается ошибка. Действительно, если анализируемая граница сдвинута во внешнюю область по сравнению с заданной на величину, большую □, то в точке C выполнится условие, анализируемое на границе, что является указанием наличия ошибки. При сдвиге границы внутрь анализируемой подобласти неправильно выполняются тесты в точках A и B или в одной из них. При таком расположении контрольных точек обнаруживаются не только ошибки вычислений, используемых в предикатах, но и ошибки операторов предикатов и маршрутов. Даже в случае ошибочной замены неравенства ≤ на < ошибка обнаруживается в результате неправильных данных в точках A и C.

Для проверки области изменения данных, образующих маршрут, необходимо тестировать граничные значения каждого предиката. В зависимости от числа ветвлений в каждом маршруте может иметься $n_в$ таких границ. Для каждой границы, как показано выше, необходимо иметь три тестовых значения. Однако число тестовых значений можно несколько сократить, если крайние из них выбирать на пересечениях соседних границ. В результате число тестов для каждого маршрута β лежит в пределах $2n_в \le \beta \le 3n_в$.

Дальнейшее сокращение числа тестовых значений возможно, если учитывать корреляцию условий на различных маршрутах, имеющих общие границы областей. В ряде случаев проверка программы в точках A и C (см. рис. 2.10) может проводиться только на одном

маршруте, соответствующем нижней части области изменения данных. На маршруте, использующем верхнюю часть области, достаточно тестировать только при тестовых значениях, соответствующих точке B.

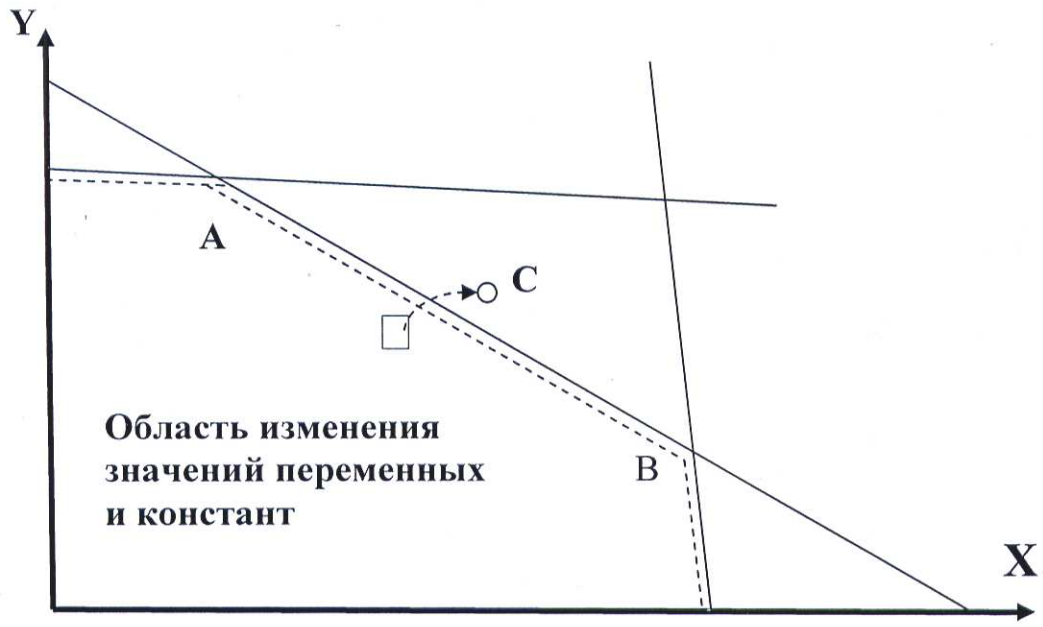


Рис. 2.10

Для полной проверки предиката ***строгого равенства*** необходимы два теста на границе условия равенства значений и два теста, имеющие малые отклонения от этой границы. Таким образом, по сравнению с проверкой неравенств в данном случае прибавляется еще один тест. При этом гарантируется обнаружение ошибок как в операторе предиката, так и в расчете данных при формировании границы анализируемого условия. Приведенный подход полностью может быть отнесен к анализу предикатов типа строгого неравенства. Проверка совокупности маршрутов, содержащих строгие неравенства и равенства, может быть сокращена, если учитывать при последовательном тестировании уже проверенные области.

При использовании стратегии областей по признаку их выпуклости могут автоматически выделяться неверные предикаты. Если одна из граней многогранника приводит к нарушению его выпуклости, то область считается аномальной и может быть локализован предикат, содержащий ошибку. Последовательный попарный анализ предикатов на их совместимость и непротиворечивость позволяет отсеивать некоторые нереализуемые маршруты. Тем самым в ряде слу-

чаев значительно сокращается необходимый объем тестирования модуля.

Приведенный метод упорядоченного регулярного тестирования на основе определения областей изменения данных является весьма эффективным. Сложность тестов линейно растет с увеличением размерности пространства исходных данных (числа переменных) и с ростом числа предикатов на маршрутах. Для многих типовых модулей сложность тестов потока данных оказывается допустимой для полной проверки модуля. Ограничения метода проверки областей могут проявляться при сложных организациях циклов, когда резко возрастает число маршрутов и анализируемых условий. Значительные трудности возникают при нелинейных предикатах. Даже нахождение точек пересечения нелинейной границы с множеством линейных границ может потребовать сложных вычислений. При использовании в программе операторов ***«или»*** необходимо установить характер перекрытия областей, соответствующих анализируемому оператору, что требует дополнительных расчетов. Тем не менее, метод анализа областей изменения данных может существенно упорядочить тестирование программ и сократить число ошибок.

Тестирование корректности определения и использования данных на маршрутах исполнения программы (см. рис. 2.9). Если маршруты исполнения программы соответствуют допустимым областям изменения входных данных, то целесообразно проверить корректность основных операций обработки данных на выделенных маршрутах. Эти проверки могут производиться путем символического анализа текста программы или накопления результатов обработки реальных данных при исполнении программы. Каждая величина на маршруте исполнения программы считывается из памяти и после использования для вычислений записывается в память ЭВМ для хранения и последующей обработки. Чередование операций чтения и записи переменных может быть нарушено в результате ошибок в программе. Для выявления таких ошибок проводится тестирование корректности записи и считывание реальных значений данных или статический анализ этих операций по тексту программы.

При ***тестировании потока данных*** выявляются все определения переменных, которые могут достигать каждой вершины графа программы по управлению. При этом под ***определением данных*** подразумеваются действия в программе, которые изменяют соответствующую величину данных. ***Использование данных*** – это операция в

выражении, которая обращается к переменной, не изменяя ее. В результате для каждой точки программы устанавливается, какие ***переменные действуют*** в этой точке, т.е. какие данные, сформированные до прихода в эту точку, используются после выхода из нее. Анализ потока данных позволяет обнаруживать ошибки, обусловленные нарушением корректной последовательности операций определение – использование данных и выявлять переменные, которые целесообразно сохранять на регистрах.

Анализ процесса определение – использование или ***«жизни данных»*** в программе естественно проводить по маршрутам ее исполнения. Для каждой единицы данных может быть установлена точка освобождения этой переменной, после которой она больше не используется и не определяется. Некоторые последовательности операций ***определение - использование*** над переменными могут быть аномальными. Так, например, последовательное изменение переменной без промежуточного использования на некотором маршруте, скорее всего, содержит ошибку в программе. Для выявления подобных аномалий необходимо упорядочить узлы графа программы так, чтобы операция в вершине не выполнялась до тех пор, пока не выполнятся все предшествующие ей узлы. При наличии циклов пока нет способов для предсказания информации, достигающей узлов, при однократном прохождении через граф. Значительные трудности для выявления аномалий представляют вызываемые процедуры. В этом случае необходимо раскрывать процесс использования данных процедурами.

Тестирование корректности и точности результатов вычислений каждой переменной. Когда показано, что сочетания данных и их области определения соответствуют корректному формированию маршрутов в программе, а также нет явных ошибок в последовательностях определения и использования каждой переменной, целесообразно провести тестирование корректности обработки каждой переменной и точности вычислительной части программы. Этот вид тестирования производится преимущественно маршрутов по областям определения.

Множество тестовых значений для проверки вычислений при простых числовых переменных целесообразно строить упорядоченно с учетом следующих правил:

- входные тестовые данные в области гладкого изменения, зависящих от них результатов, должны принимать, по крайней мере,

значения, близкие к наибольшим и наименьшим, а также одно – два промежуточных значения;

- тестирование должно проводиться при всех особых значениях входных переменных в точках резкого возрастания или разрыва производных, при нулевом, единичном и предельно малых численных значениях;
- входные тестовые значения должны обеспечивать проверку программы при выходных результатах, имеющих особые точки резкого изменения или разрыва производных;
- если значения некоторой переменной зависят от значений другой переменной, то необходимо тестировать при особых значениях сочетаний переменных (равенство обеих переменных, малое и предельно большое их различие, нулевые и единичные значения).

Таким образом, для каждой простой числовой переменной, кроме трех точек вблизи и на границе области определения, обычно необходимо тестирование программы в 3 – 4 промежуточных и в 2 – 5 особых точках значений входных данных. При 10 входных переменных и сложных вычислениях в программном модуле для тестирования вычислений может потребоваться до 50 тестовых значений. Группируя и упорядочивая тестовые значения разных переменных, их общее количество можно сократить до 5 – 10 тестовых наборов.

В вычислительных программах переменные часто представлены в виде массивов. При наличии массивов объем тестирования значительно увеличивается. Для проведения эффективного упорядоченного тестирования ***при переменных образующих массив***, необходимо учитывать:

- структуру и размер массива, а также наличие в структуре массива особых точек или подструктур;
- изменение области определения и особых точек значений каждой переменной при расположении данных в различных местах массива.

В простейшем случае (отсутствие особых точек в одномерной структуре массива и в значениях переменной) при тестировании необходимо минимум 3 значения переменной (краевые и промежуточные) при ее расположении в 3 точках массива, т.е. 9 тестов. При увеличении размерности массива и появлении особых точек в его структуре или значениях переменной соответственно возрастает число тестов, необходимых для проверки программы. Если область определения переменной включает значение нуль, то целесообразно тестиро-

вание программы при всех значениях переменной в массиве, равных нулю. Кроме того, следует задавать значения переменных в массиве равными между собой, а также изменяющимися случайно или в соответствии с наиболее характерной закономерностью. В ряде массивов можно выделить определенные столбцы, строки или иные подструктуры, имеющие самостоятельное назначение. Такой массив может быть использован для реализации иерархии подструктур. В этом случае каждая структура делится на подструктуры более низкого уровня, пока не получатся подструктуры, состоящие из отдельных элементов массива.

При формировании тестов целесообразно задавать их значения в соответствии с предельными размерами подструктуры, а также особые и типовые значения элементов в подструктуре. При этом компоненты остальных подструктур могут быть зафиксированными на некоторых типовых значениях. Затем может потребоваться перебор сочетаний типовых и особых значений каждой подструктуры с типовыми и особыми значениями размеров и элементов остальных подструктур. Перебор значений может приводить к весьма резкому росту объема тестирования. В этом случае для уменьшения объема тестирования целесообразен детальный анализ типовых особых точек реальных массивов и их структур. Выделение и упорядочение особых точек, учет симметрии подструктур и их компонент позволяют отойти от полного перебора сочетаний тестовых значений и достичь высокого качества тестирования при разумном его объеме.

Определение нереализуемых маршрутов потоков данных программных модулей (см. рис. 2.9). Предшествующее изложение базировалось на анализе отдельных условий и предикатов ***без учета их взаимосвязи при последовательном исполнении операторов*** программы. Оценка сложности тестирования модулей при этих предположениях ориентирует на наибольшие затраты. В реальных программных модулях многие предикаты, определяющие последовательные ветвления на маршрутах программы, являются взаимозависимыми и, в частности, несовместимыми. Такие маршруты, выделенные формально по полному графу программы, не могут быть исполнены при реальном функционировании программы ни при каких сочетаниях исходных данных. В результате сокращается общее число маршрутов, характеризующих данный программный модуль, и уменьшается сложность его тестирования. Это может приводить к

целесообразности изменения стратегий тестирования и к сокращению ресурсов, необходимых для достижения заданной корректности программ.

За счет нереализуемых маршрутов при любом критерии их выделения сокращается их общее число M. Если не учитывать это изменение M, то рассчитанная вероятность проявления ошибки всегда будет завышенной и ни при каком объеме тестирования не достигнет нуля. В действительности исчерпывающее тестирование при критериях f возможно только по реализуемым маршрутам. Поэтому при расчете вероятности проявления ошибки целесообразно проводить предварительную оценку числа нереализуемых маршрутов и корректировать используемое значение M.

На маршрутах графа программы несовместимые условия, исключающие возможность их исполнения, в целом разделены рядом точек (узлов)ветвления, в которых используются другие предикаты при определении направления движения по маршруту. Для не реализуемости маршрута достаточно двух несовместимых условий, которые встречаются в следующих видах:

- тождественные выражения, определяющие явно условия ветвления, которые анализируются на формально выделяемом маршруте при противоположных значениях условий движения по этому маршруту;
- выражения с одними и теми же переменными, но с разными их значениями, определяющими непересекающиеся области условий ветвления;
- выражения с различными переменными, связанные неявно логикой решаемой задачи, которая формально не отражена в данной программе.

Для ***выявления нереализуемых маршрутов*** в формально построенном графе программы, каждый из них должен быть проанализирован на наличие подобных пар несовместимых условий. Такой анализ относительно просто выполнить для условий первого вида, которые, однако не так часто встречаются. Наиболее важный вид неявных несовместимых условий формально проанализировать невозможно без привлечения описаний всех корреляционных связей между переменными и предикатами. Построение таких описаний громоздко и трудоемко, что затрудняет их применение для селекции нереализуемых маршрутов. Выделение непересекающихся областей значений переменных при формировании условий образования маршрутов

трудно реализовать из-за часто встречающихся областей со сложной конфигурацией. Дополнительные трудности может представлять селекция условий с точными и неточными неравенствами на границах областей, когда граничные значения по-разному используются в анализируемых условиях маршрута (см. рис. 2.9).

Приведенные факторы затрудняют формализованное выделение нереализуемых маршрутов и оценку влияния несовместимости условий на снижение сложности тестирования модулей. Особенно быстро возрастают трудности при попытке формального выделения нереализуемых маршрутов в группах программ, состоящих из многих модулей. Эти данные получались либо в процессе ручного анализа предикатов на маршрутах, построенных в результате автоматического формального анализа графов программ, либо путем набора статистики числа различных исполняемых маршрутов при реальном функционировании программ на случайных исходных данных.

Достаточно часто встречаются модули, в которых все маршруты являются реализуемыми и отсутствуют несовместимые условия при ветвлениях. В ряде программ несовместимые условия могут приводить к сокращению числа маршрутов в несколько раз. Анализ некоторых логических программ показал, что доля нереализуемых маршрутов из их общего числа может быть весьма велика и достигает в отдельных случаях 90%. При этом доля нереализуемых маршрутов в среднем быстро возрастает при увеличении числа ветвлений в программе. Это обусловлено тем, что вероятность не реализуемости маршрута зависит от его длины, точнее, от числа ветвлений. На коротких маршрутах с небольшим числом ветвлений, естественно, менее вероятно наличие одинаковых предикатов, определяющих ветвления, и вероятность не реализуемости низка. В пределе маршруты с одним – двумя ветвлениями всегда являются реализуемыми.

Особенно значительное снижение сложности тестирования за счет исключения нереализуемых маршрутов происходит в программных модулях с циклами. Например, циклы, имеющие фиксированный параметр выхода из цикла, образуют нереализуемые маршруты при всех остальных значениях параметра. Наличие в теле цикла нескольких предикатов и их (обычно неявная) логическая связь, а также корреляция с номером витка исполняемого цикла, приводит к тому, что при тестировании нет необходимости полного перебора всех условий в теле цикла на каждом его витке. Путем ручного неформального

анализа в некоторых случаях удается эффективно выделять реализуемые маршруты с циклами, без анализа всех сочетаний несовместимых условий, при которых образуются нереализуемые маршруты. Автоматизация выделения нереализуемых маршрутов ограничена технической сложностью переборного анализа предикатов на маршрутах и принципиальной сложностью идентификации несовместимых условий.

Документы при тестировании программных модулей

Регламентировать тестирование программных модулей должны следующие базовые документы:

- исходные данные для тестирования модулей;
- организация, подготовка тестирования и обеспечение качества модулей;
- сценарии тестирования и спецификации тестов для каждого модуля;
- отчеты о результатах верификации и тестирования модулей.

Документы тестирования должны содержать исходные тексты запрограммированных и оформленных модулей и описаний данных, общий план организации и порядок их тестирования. В отчетных документах должны отражаться результаты тестирования компонента и допустимость его применения в комплексе программ.

Исходные данные для тестирования модулей должны включать следующую документацию:

- техническое задание и/или спецификация требований на разработку программного модуля;
- исходный текст программы в виде печатного документа и файла;
- выбранные методы тестирования модулей;
- эталонные значения и/или распределения исходных и результирующих данных, отражающие требующиеся функции, сценарии тестирования и покрытие модуля тестами в заданном разнообразии его поведения;
- тестовые сценарии исходных и результирующих данных, являющиеся достаточно представительной выборкой из полной совокупности значений и распределений эталонных данных, ограниченной доступными ресурсами на тестирование модуля;

- доступные ресурсы для тестирования модуля: финансовые, трудовые и временные затраты на тестирование; оснащенность процесса тестирования модуля программными и аппаратными средствами автоматизации; состав и квалификация специалистов;
- критерии качества результатов тестирования и требуемая полнота покрытия тестами модулей, в которые входят описания реализуемых функций и характеристики качества реализации функций, в пределах которых считается, что модуль удовлетворяет предъявленным требованиям к функциям и качеству.

Организация, подготовка тестирования и обеспечение качества модулей должны представляться в документе:

- методы, технология и средства автоматизации разработки и тестирования, обеспечивающие создание каждого модуля с заданными показателями качества;
- методы и средства объективного измерения требуемого покрытия тестами и достигнутого качества результатов тестирования каждого модуля;
- документы и методики для обеспечения и контроля соблюдения правил и технологии проектирования, тестирования и обеспечения всего жизненного цикла модулей;
- структура и содержание (шаблон) каждого документа отражающего результаты этапа тестирования модуля, качество выполненных работ и полноты покрытия тестами.

Сценарии тестирования и спецификации тестов для каждого модуля должны отражаться в документе:

- метод и вид тестирования адекватный модулю, а также основной цели его выполнения;
- план тестирования в соответствии с выбранным методом с учетом ограниченных ресурсов испытаний, имеющихся для достижения заданной достоверности проверки качества модуля;
- спецификация общей схемы сценариев тестирования модуля, критерии, позволяющие его оценивать;
- описания контрольных сценариев тестирования – набор конкретных тестовых значений и соответствующих им эталонов, ожидаемые, эталонные выходные значения результатов тестирования;
- результаты функционирования модуля при подготовленных тестах и заданиях;

- сравнения результатов тестирования с эталонами и обнаруженные отклонения (дефекты или ошибки) для проведения дополнительного тестирования с целью диагностики и локализации дефектов;
- верификация корректности интерфейсов взаимодействия модуля в составе программного компонента должна удовлетворять исходным системным требованиям, утверждена организационная ответственность внутри процесса верификации компонентов и интерфейсы с другими процессами жизненного цикла.
- оценки полноты проведенного тестирования, степени покрытия модуля тестами выбранным методом и необходимости применения других методов и видов тестирования для увеличения покрытия тестами;
- оценка характеристик качества модуля, исходных и выходных данных: годен - не годен;
- оценки наличия ресурсов для продолжения тестирования и момента его завершения, а также для определения достигнутых характеристик качества модуля или выбора очередного метода тестирования.

Отчет о результатах верификации и тестирования модулей:

- справка о передаче модуля на тестирование, когда в разработке участвуют независимые группы программистов и тестировщиков;
- журнал выполнения плана тестирования, используемый коллективом тестировщиков модулей для регистрации: сценариев и операций, которые имели место во время выполнения тестирования; аномальных событий и дефектов для диагностики и локализации причин аномалий; изменений и корректировок, которые произведены в модулях;
- результаты верификации и тестирования: результат выполнения (прошел/не прошел) для каждого анализа и выполненного теста и заключительный результат верификации и тестирования модуля; результаты оценивания покрытия и анализа трассируемости для выполнения тестов, и анализов в процессе верификации и тестирования;
- итоговый обобщенный отчет верификации и тестирования модулей; результаты работ по верификации и тестированию, сценариев или видов проверки компонентов и комплекса в целом;
- модули и компоненты комплекса программ выполняют все (частично) декларируемые в документации функции;

- полученные выходные данные находятся в допустимых пределах отклонений от эталонных, заданных в требованиях на разработку комплекса программ или спецификациях на модули;
- программная документация соответствует требованиям стандартов.

Затраты на производство программных модулей и компонентов

При тестировании модулей и компонентов комплексов программ основным лимитирующим ресурсом обычно являются допустимые трудозатраты специалистов, а также ограничения на сроки разработки версии программного комплекса, на параметры ЭВМ и технологию тестирования. Одним из наиболее важных составляющих планирования тестирования является оценка трудоемкости и времени, необходимых для его выполнения. Затраты на тестирование требований компонентов программных комплексов могут составлять существенную часть (до 30%) стоимости проекта, при этом жизненно важно для успеха этой операции, чтобы тестирование проводило достаточное число специалистов и у них было достаточно времени на качественное выполнение задач по тестированию и корректировкам программ. Ограничения реальных ресурсов на верификацию и тестирование компонентов определяют достижимое качество версий комплексов программных продуктов.

Затраты на тестирование программных модулей зависят от их индивидуальной сложности и в одном комплексе программ могут различаться во много раз. ***Ориентиром для оценки относительной трудоемкости тестирования*** определенного модуля может служить число узлов и сложность циклов в графе совмещенного потока управления и потока данных. Простой подсчет числа строк в тексте программы модулей можно использовать только для первичной оценки при распределении ресурсов на новый комплекс программ. В этих случаях при планировании производства модулей сложных программных комплексов целесообразно использовать экспериментальные статистические распределения основных экономических характеристик – трудоемкости, длительности и числа специалистов ***по этапам работ и по реальному времени производства компонентов***. Относительные значения распределения этих величин на интервале реализации сложных программных комплексов значительно раз-

личаются в зависимости от размера и типа комплекса программ, а также от числа модулей и компонентов в комплексе. Однако общие тенденции состоят в наименьших затратах на начальных и конечных этапах производства, и в наибольших суммарных затратах, на средних этапах, когда создаются новые программные компоненты и модули.

В совокупных затратах на производство полностью новых модулей и комплексов программ доминирует ***трудоемкость непосредственной разработки, программирования и тестирования программных модулей***. Наиболее полно эти данные исследованы и обобщены для двух классов комплексов программ: для встроенных комплексов программ в системах реального времени (СРВ) и для полунезависимых программ административных и информационно-поисковых систем (АС). Для этих распределений характерно наличие максимума в использовании трудовых ресурсов на средних этапах разработки – на этапах ***программирования и тестирования модулей***. Эти этапы полностью заняты производством компонентов, а на остальных этапах к ним относится только часть затрат. После выделения затрат на создание модулей, остающиеся затраты можно отнести к процессам проектирования, а также сборки – интегрирования компонентов и испытаний комплексов программ.

Экспериментальные распределения относительного числа занятых специалистов при производстве сложных комплексов программ этих классов одинакового размера подобны и положения экстремумов в этих распределениях различаются относительно мало. Программы модулей класса СРВ зачастую имеют существенно ***большую трудоемкость***, чем для класса АС, и особенно различаются на завершающих этапах комплексирования и испытаний. Повышенные затраты и число участвующих специалистов для класса СРВ характерны на этапах программирования и тестирования модулей вследствие более жестких требований к их качеству. В результате относительное число специалистов при разработке комплексов программ СРВ меньше на начальных и конечных этапах. Однако вследствие большой трудоемкости полное число необходимых специалистов при разработке программ СРВ значительно больше, чем при создании класса АС.

В общих оценках доли затрат на программирование и тестирование комплексов программ трудно выделять и оценивать затраты на

создание конкретных модулей, вследствие их оригинальности и неповторимости. Априорные оценки затрат на модуль по числу узлов и циклов весьма трудоемки, хотя и достаточно достоверны. Однако для определенных предприятий и классов программ при завершении их проектов полезно проводить оценки средней трудоемкости, длительности и числа программистов, участвовавших в создании одного модуля. Эти величины можно получать путем регистрации интегральных затрат на программирование и тестирование программных модулей и деления их значений на количество созданных модулей. При последующем создании подобных проектов полученные средние затраты на модуль могут служить ориентирами при первичном планировании нового комплекса программ.

Прогнозирование затрат на тестирование программных модулей и компонентов, более или менее корректно, возможно на основе обобщения статистических данных предшествующих проектов. В данном случае, задача ограничена только ориентировочным перечнем основных составляющих затрат, которые целесообразно учитывать в процессе тестирования. Этот перечень может использоваться как ***ориентир*** при подготовке Программы тестирования компонентов. Обычно наиболее важным для реализации проекта и зависящим от большинства его особенностей и факторов является ***трудоемкость, непосредственно определяющая стоимость*** тестирования создаваемого компонента программ. Значения длительности разработки и числа специалистов взаимосвязаны и в некоторых пределах могут разменяваться. Поэтому оценки этих показателей затрат можно варьировать, и при недостаточном числе специалистов естественно возрастает длительность разработки, хотя трудоемкость может остаться практически неизменной. Многократное применение одних и тех же апробированных компонентов и/или их адаптация к различным условиям применения является одним из ***перспективных методов повышения качества и снижения затрат труда специалистов*** на тестирование модулей и компонентов комплексов программ.

Оценки трудозатрат на тестирование могут составлять существенную часть стоимости проекта, при этом жизненно важно для успеха, чтобы тестирование компонентов и модулей проводило достаточное число специалистов, и у них было достаточно времени на качественное выполнение своих задач. Получение ***оценки*** трудоза-

трат на компоненты комплекса программ целесообразно разбить на ***этапы*** (см. рис. 2.8) **.**

Определение перечня и состава задач и компонентов тестирования, которые должны быть выполнены. Эта оценка начинается с определения работ, которые необходимо выполнить для того, чтобы тестирование компонента считалось состоявшимся. На этом этапе может быть достаточным разбить работу на функциональные задачи, отражающие проверку реализации конкретных требований к функциям и характеристикам компонентов комплекса программ. Если используются менее формальные методы, то результатом этого этапа может быть простой список основных задач, модулей и программных компонентов.

Оценка трудозатрат на решение отдельных задач и процесс тестирования модулей и компонентов всего комплекса программ. Каждая задача и компонент, выявленные на первом этапе, требуют для своего решения определенных трудозатрат, представляющих собой объем работ, необходимых для выполнения соответствующей задачи. Оценки этих трудозатрат могут быть представлены в виде произведения количества исполнителей на затраченное ими время и измеряться в таких единицах, как человеко-день или человеко-месяц.

Определение времени, требуемого для решения каждой задачи и длительности тестирования компонентов. Время, необходимое для решения задачи, измеряется в днях, неделях или месяцах. Время, необходимое для выполнения той или иной задачи, зависит от количества исполнителей, однако эта зависимость не обязательно линейная. Суммарная продолжительность работ по тестированию компонентов зависит от продолжительности решения отдельных функциональных задач, однако это не простое суммирование, поскольку некоторые задачи можно решать параллельно и одновременно с другими.

Оценка рисков невыполнения графика работ и формулировка планов их снижения. Следует оценивать возможные проблемы и риски, которые могут возникнуть при решении задач в запланированные промежутки времени, и предусмотреть средства решения этих проблем. В самом начале проекта (перед выявлением и изучением требований) очень трудно определить, какой окажется стоимость проекта, и в частности длительность тестирования его компонентов. Фактическая стоимость проекта комплекса программ становится известной только на завершающей стадии проекта. На начальной стадии проектирования может иметь место как недооценка, так и переоценка фак-

тической стоимости проекта. После того, как все требования будут проанализированы и утверждены, оценка стоимости проекта обычно отличается от окончательной, фактической стоимости примерно в два раза.

Затраты на обеспечение корректности функциональных компонентов и модулей комплекса программ зависят от полноты прослеживания реализации требований к ним сверху вниз. Эти затраты входят непосредственно в процесс разработки и зависят от объема и детальности ***процессов верификации и тестирования требований к компонентам и модулям***. Для сложных программных комплексов при требовании их высокой корректности они могут составлять до 25 – 30% от затрат труда и 15 – 20% времени на обеспечение первичной, функциональной пригодности. Эти затраты приходятся на верификацию и тестирование программных компонентов и модулей, что должно обеспечивать корректность, безопасность и надежность комплекса в целом.

Затраты на обнаружение и устранение дефектов и ошибок в программе определяются двумя факторами: затратами на обнаружение каждой ошибки и затратами на устранение выявленных ошибок при формировании очередной версии. Чем меньше ошибок в программе, тем труднее они обнаруживаются, т.е. тем выше затраты на выявление каждой ошибки. Затраты на устранение ошибок и корректировку программ пропорциональны числу дефектов, выявляемых между очередными версиями программного компонента. По опыту работ, широко тиражируемый комплекс программ объемом ~ 10^5 строк, может требовать непрерывных усилий коллектива в составе десятка и более специалистов для устранения ошибок, корректировок версий и документации.

Затраты на совершенствование и модернизацию комплексов программ близки по содержанию (но не по величине) к затратам на их первичную разработку. Модернизация обычно производится поэтапно. Для каждой новой версии изменяется (разрабатывается) только некоторая часть от всего объема программного комплекса. Экспериментально установлено, что эта часть при вводе очередной версии обычно составляет 10 – 20% от объема всего комплекса. Сложность связей компонентов приводит к тому, что удельные затраты на изменяемые программы при модернизации каждой версии могут быть в 2 – 3 раза больше, чем затраты на создание модулей программ ***тако-***

го же объема при разработке первой версии. Эта величина зависит от того, насколько путем стандартизации архитектуры и интерфейсов, предусматривались перспективы совершенствования комплекса программ.

Лекция 2.3

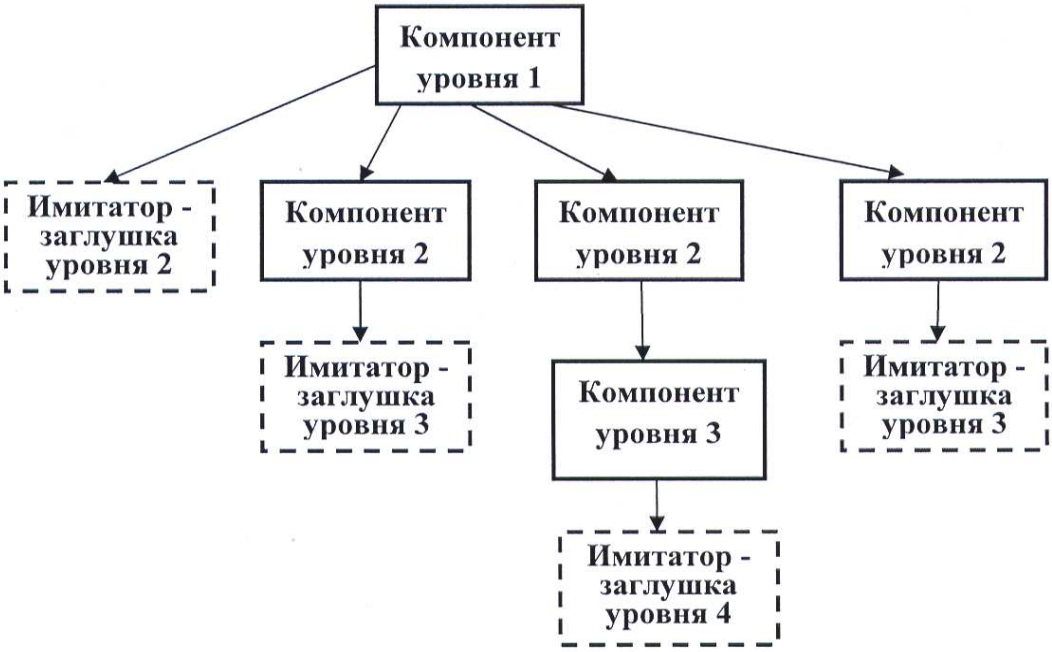
ПЛАНИРОВАНИЕ ТЕСТИРОВАНИЯ МОДУЛЕЙ И КОМПОНЕНТОВ ДЛЯ КОМПЛЕКСА ПРОГРАММ

Нисходящая – восходящая сборка и тестирование модулей и программных компонентов

Проектирование и выполнение тестов для групп модулей остаются такими же, как и для моделей потока управления, с некоторыми различиями, которые определяются графами потока данных, а не только графами потока управления. В дальнейшем для простоты предполагается, что в модулях нет циклов. Сборка комплексов программ из модулей и компонентов занимает важное место при создании сложных программных продуктов. В зависимости от роли и места сборки в общем технологическом процессе создания комплексов программ различаются два метода тестирования: ***нисходящий*** и ***восходящий*** (рис. 2.11). Соответственно при нисходящей интеграции компоненты высокого уровня интегрируются и тестируются еще до окончания проектирования, комплексирования и реализации комплекса программ. При восходящей интеграции перед разработкой и сборкой компонентов более высокого уровня сначала интегрируются и тестируются модули и компоненты нижнего уровня.

Нисходящий технологический процесс – основан на сочетании пошаговой детализации технического задания на последовательных этапах проектирования с её завершением на таком уровне, когда проект может быть описан из готовых компонентов и модулей, хранящихся в базе данных ***«сборочного программирования»***. На основании требований технического задания и описания проблемной области создается параметризованное функциональное описание разрабатываемого комплекса программ. При проектировании осуществляется последовательная разработка спецификаций требований к создаваемому комплексу к его составным компонентам и модулям – рис. 2.12.

Нисходящее тестирование сборки компонентов



Восходящее тестирование сборки компонентов

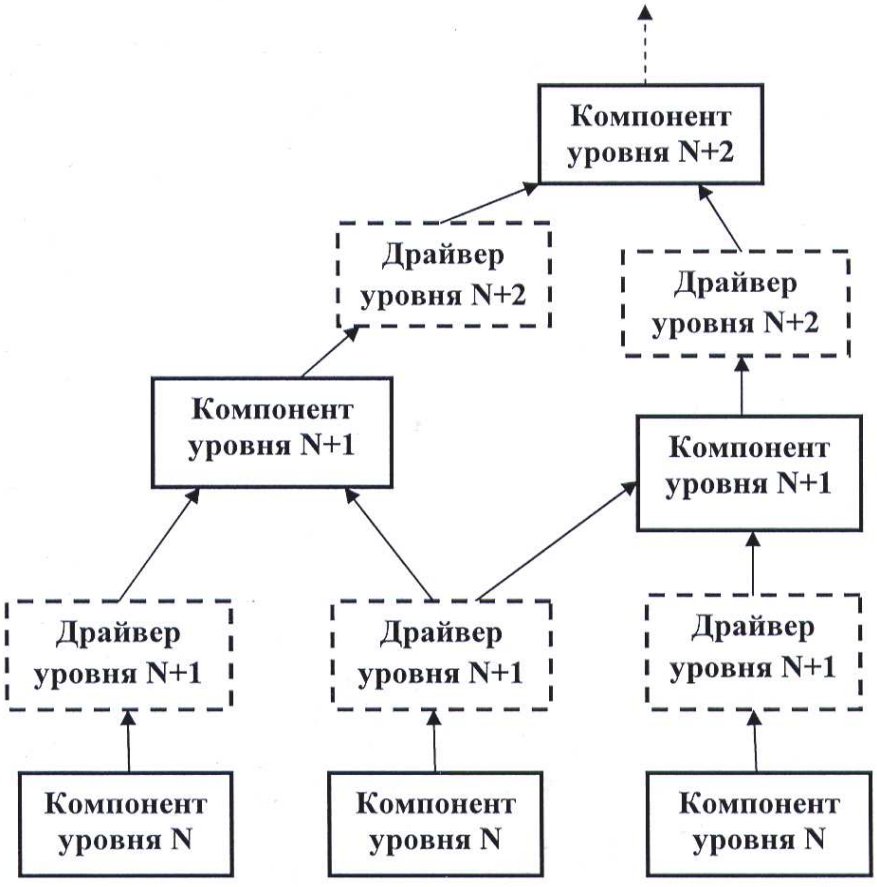


Рис. 2.11

Планирование тестирования модулей и компонентов для комплекса программ должно включать:

- выбор метода «нисходящей – восходящей» сборки и тестирования модулей и программных компонентов:
  - планирование порядка разработки программ модулей и компонентов и их тестирования для комплекса программ;
  - формирование требований и эталонов к тестам в базе данных и/или матрице для отслеживания процессов реализации требований;
  - тестирование интерфейсов модулей и программных компонентов;
- планирование тестирования модулей и компонентов для комплекса программ:
  - формирование предварительного плана разработки и порядка выполнения тестов для модулей и компонентов комплекса программ;
  - документирование квалификации сотрудников, необходимых для тестирования модулей и компонентов;
  - формирование и контроль матрицы отслеживания требований к тестам модулей и компонентов;
  - выбор, подготовка и утверждение детализированных планов производства и тестирования программных модулей и компонентов;
  - анализ и контроль покрытия тестами требований и эталонов при тестировании модулей и компонентов;
- подготовка графиков разработки и выполнения тестов для модулей и компонентов комплекса программ;
  - освоение методов формирования графиков производства модулей и компонентов для программных комплексов;
  - стандартизация планирования производства комплекса программ и подготовки детализированных планов для модулей и компонентов;
  - отслеживание изменений требований и тестов и соответствующее обновление графиков тестирования;
  - тестирование качества технологической документации на модули и компоненты комплекса программ;
- применения графиков для планирования производства компонентов и комплексов программ:
  - освоение методики и средств планирования производства и тестирования компонентов сложных комплексов программ с использованием графиков Ганта;
  - принципы построения и применения сетевых графиков для планирования производства компонентов и комплексов программ.

Рис. 2.12

При построении версий из повторно используемых компонентов (ПИК) может не оказаться элементов, полностью подходящих по функциональным и программным признакам. В этом слу-

чае недостающие части модулей разрабатываются как ПИК с тем, чтобы конструктивно-технологические требования ко всем составным частям комплекса соответствовали типовой структуре, выработанной для всей проблемной области. При наличии ограничений осуществляется контроль ПИК и создаваемых компонентов по конструктивным показателям (емкость памяти, интерфейсы, время функционирования). По завершении этого этапа в инструментальном комплексе должны содержаться описания (спецификации) всех программных и информационных модулей и компонентов. Эти компоненты должны быть согласованы в функциональном и структурном плане и по конструктивно-технологическим параметрам. В результате может быть подготовлена автоматизированная сборка комплекса программ.

Нисходящее тестирование – является неотъемлемой частью процесса нисходящей разработки сложных программных комплексов, при котором сначала разрабатываются и тестируются компоненты верхнего уровня, а затем компоненты, находящиеся на нижних уровнях иерархии. При наличии не полного состава готовых ПИК комплекс можно представить в виде одного абстрактного компонента с упрощенными, временными функциональными заглушками, которые последовательно заменяются на реальные модули и компоненты. ***Заглушки-имитаторы*** должны иметь такой же интерфейс, что и готовые ПИК, но с ограниченной функциональностью. После того как компоненты верхнего уровня запрограммированы и протестированы, таким же образом реализуются и тестируются его субкомпоненты. Процесс продолжается до тех пор, пока не будут полностью реализованы компоненты и модули самого нижнего уровня. Затем весь комплекс программ и система тестируются целиком.

Восходящий технологический процесс – применяют в тех случаях, когда объем комплекса не очень большой, разработчики хорошо знают структуру и имеют весь состав готовых модулей и компонентов. При этом исходят из того, что существует ***прототип*** создаваемого комплекса, соглашения, принятые, при разработке прототипа, распространяются и на новый программный комплекс. Предполагается функциональное подобие нового комплекса и прототипа, разработка осуществляется итерационным путем. Вначале оценивают состав и качество имеющихся готовых ПИК, из них планируют собрать версию, выбирают направления и объекты для последовательных доработок. Доработки, как правило, касают-

ся изменения и совершенствования функциональных возможностей, интерфейсов разных видов при смене оборудования или операционной системы, изменений эргономических требований или модификации дисциплины обмена с абонентами сети или вычислительной системы. Доработки не всегда удается сделать так, чтобы полностью удовлетворить требованиям комплекса программ. Поэтому по завершении доработок в какой-то части комплекса осуществляется повторная его сборка и оценка полноты реализации требований технического задания. Этот цикл работ может повторяться в зависимости от соотношения потребных и завершенных доработок компонентов и модулей, особенно в условиях сжатых сроков разработки. Итерации, завершающиеся сборкой, осуществляют для распараллеливания работ по доработке компонентов и по комплексному тестированию предварительного варианта группы компонентов и модулей. Восходящий технологический процесс применяют обычно в коллективах, в течение ряда лет разрабатывающих однотипные комплексы программ для разных заказчиков.

При ***восходящем тестировании*** – сначала интегрируются и тестируются модули и компоненты, расположенные на низких уровнях структурной иерархии. Затем выполняется сборка и тестирование модулей, расположенных на верхнем уровне иерархии, и так до тех пор, пока не будет протестирован последний компонент и весь комплекс. При таком подходе не требуется наличие законченного архитектурного проекта комплекса, и поэтому он может начинаться на раннем этапе разработки. Обычно такой подход применяется тогда, когда в системе есть почти полный состав повторно используемых компонентов.

При восходящем технологическом процессе по завершении испытаний программного продукта полезно осуществлять отбор компонентов, которые могут рассматриваться как повторно используемые, а также ввод их в базу данных испытанных гарантированных ПИК. В качестве нового может рассматриваться компонент, полученный в процессе доработки из ранее гарантированного ПИК. В этом случае он отличается от старого либо расширенными функциями и областью определения параметров, либо учетом какого-либо нового вида интерфейса или устройства. Вопрос о целесообразности замены в ранее отлаженных комплексах ПИК на новый, доработанный, следует решать с большой осторожностью, особенно при использовании ПИК в комплексах, сильно связан-

ных по информации (например, в системах реального времени). В результате включение нового ПИК вместо старого может приводить к необходимости повторной комплексной отладки всего программного комплекса.

Преимущества нисходящего и восходящего тестирование можно сопоставлять с учетом ***следующих факторов***.

Верификация и аттестация системной архитектуры при нисходящем тестировании имеет больше возможностей выявить ошибки в архитектуре и требованиях к комплексу программ на раннем этапе разработки. Обычно это структурные и функциональные ошибки, раннее выявление которых предполагает их исправление почти без дополнительных затрат. При восходящем тестировании структура высокого уровня не утверждается вплоть до последнего этапа разработки всего комплекса программ.

Демонстрация комплекса программ может быть возможна при нисходящей разработке уже на ранних этапах разработки. Этот факт является важным психологическим стимулом использования нисходящей модели разработки программных комплексов, поскольку демонстрирует заказчику осуществимость всей управляющей системы. Аттестация проводится в начале процесса тестирования путем создания ***демонстрационной версии комплекса*** (с заглушками), но если система создается из повторно используемых компонентов, то и при восходящей разработке также можно создать ее демонстрационную версию.

Сложность реализации тестов выше при нисходящем тестировании, так как необходимо моделировать программы-заглушки компонентов и модулей нижних уровней. ***Программы-заглушки*** могут быть упрощенными функциями версий представляемых компонентов. При восходящем тестировании для того, чтобы использовать компоненты нижних уровней, необходимо разработать ***тестовые драйверы***, которые эмулируют окружение компонента или модуля в процессе тестирования.

Наблюдение за ходом тестирования – при нисходящем и восходящем тестировании могут возникать проблемы, связанные с контролем за состоянием тестирования. В большинстве комплексов, разрабатываемых сверху вниз, более верхние уровни системы, которые реализованы первыми, не генерируют выходные данные, однако для проверки этих уровней нужны какие-либо имитированные выходные результаты. Испытатель должен создать искусственную среду для

генерации результатов теста. При восходящем тестировании также может возникнуть необходимость в создании искусственной среды (***тестовых драйверов***) для исследования функционирования компонентов и модулей нижних уровней.

На практике при разработке и тестировании сложных систем чаще всего используется ***композиция восходящих и нисходящих методов***. Разные сроки разработки для разных компонентов предполагают, что группа, проводящая тестирование и интеграцию, должна работать с какими-либо готовыми компонентами. Поэтому во время процесса тестирования сборки в любом случае приходится разрабатывать как заглушки, так и тестовые драйверы.

Завершающим этапом при нисходящем технологическом процессе является ***отбор и наполнение базы данных предприятия новыми высококачественными апробированными компонентами и модулями***. Этот этап важен не только как поставляющий новые ПИК, но и как методическая основа для обучения специалистов современным методам разработки комплексов программ. Такая стратегия рассчитана на несколько лет, на которую ***должно ориентироваться*** предприятие, осуществляющее разработку программных комплексов в некоторой проблемной области. Поэтому требуются значительные начальные затраты на создание нормативно-технической базы, наращивание аппаратной и программной оснащенности, обучение специалистов, с тем чтобы длительно обеспечить существенный рост производительности труда и значительное сокращение сроков производства программных комплексов. Совокупность организационно-технических мероприятий по реализации «сборочного программирования» должна обеспечивать высокую стабильность технико-экономических показателей по всем планируемым разработкам.

Тестирование интерфейсов должно выполняться в тех случаях, когда модули или компоненты интегрируются в комплексы. Каждый модуль или компонент, который вызывается другими компонентами комплекса, имеет заданный интерфейс. Цель тестирования интерфейсов – выявить ошибки, возникающие в комплексе при неправильных предположениях о используемых интерфейсах. Контрольные тесты применяются не только к отдельным компонентам, но и к функциональным подсистемам, полученным в результате комбинирования и взаимодействия компонентов. Обычно статические методы тести-

рования более рентабельны, чем специальное автономное тестирование интерфейсов.

Тестирование и обнаружение дефектов интерфейсов сложно, поскольку некоторые ошибки могут проявиться только изредка при необычных условиях. Такую ситуацию можно обнаружить только во время выполнения специальных тестов: например, специально вызывается переполнение очереди, которое приводит к непредсказуемому поведению объекта. Другая проблема может возникнуть из-за взаимодействий между ошибками в разных программных модулях или компонентах. Ошибки в компоненте иногда можно выявить, когда поведение другого взаимодействующего компонента становится непредсказуемым. Если объект неправильно понимает вычисленные значения, возвращаемое значение может быть достоверным, но неправильным во времени.

Накопленный опыт, анализ дефектов и ошибок взаимодействия компонентов сложных комплексов программ позволил представить некоторые полезные ***рекомендации при тестировании интерфейсов***:

- составить список всех вызовов, направленных к внешним компонентам и такие наборы тестовых данных, при которых параметры, передаваемые внешним компонентам, принимают крайние значения из диапазонов их допустимых значений – использование ***экстремальных значений параметров*** может с высокой вероятностью обнаруживать несоответствия в интерфейсах;
- при вызове компонента через процедурный интерфейс используются тесты, вызывающие сбой в работе компонента – распространенная причина ошибок в интерфейсе, ***неправильное понимание спецификации компонентов***;
- в системах передачи сообщений используются динамические ***тесты с перегрузкой***, генерирующие большее количество сообщений, чем при обычной работе системы, эти же тесты позволяют обнаруживать дефекты синхронизации взаимодействия компонентов;
- при взаимодействии нескольких компонентов через общую память разрабатываются тесты, которые изменяют ***порядок активизации компонентов***, с их помощью можно выявить ошибки, сделанные программистом вследствие неявных предположений о порядке использования компонентами разделяемых данных.

Планирование тестирования модулей и компонентов для комплекса программ

Тестирование компонентов и сложного комплекса программ составляет около половины затрат на полное проектирование и производство программного комплекса. Составление графика производства – ***одна из ответственных работ***, выполняемых менеджером, необходимая для определения и контроля интегральных ***характеристик производства программных комплексов***. Менеджер должен оценивать длительность этапов создания компонентов и всего проекта, определять виды и размер ресурсов, необходимых для реализации отдельных этапов и типов работ, и представлять их в виде согласованной последовательности. Если данный проект подобен ранее реализованному, то график производства нового продукта можно взять за основу. Если проект является инновационным, первоначальные оценки длительности и требуемых ресурсов для компонентов почти наверняка будут ***слишком оптимистичными***, даже если менеджер попытается предусмотреть все возможные неожиданности. С этой точки зрения производство программных комплексов не отличаются от больших инновационных технических проектов, в которых также неожиданно возникают проблемы и трудности. Именно поэтому графики работ являются динамичными и их необходимо постоянно обновлять по мере поступления новой информации о ходе выполнения проекта.

Процесс планирования должен начинаться с определения ***проектных ограничений*** (временных ограничений, предельных возможностей и число специалистов, бюджетных ограничений). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как размер и структура проекта, а также при распределении функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты и документы (компоненты, подсистемы или версии программного комплекса) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается план работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого проводится контроль выполнения работ, и отмечаются расхождения между реальным и плановым ходом работ. По мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок

плана и параметров всего проекта. Это, в свою очередь, может привести к изменению графика работ.

При оценке затрат на тестирование тест-менеджер и специалисты тестирования, которые помогают ему провести оценки, должны учитывать, насколько каждый из факторов влияет на результаты. Пренебрежение или пропуск одного из факторов может превратить реалистичную оценку в неверную и бесполезную. Некоторые из этих факторов могут увеличить, или сократить сроки тестирования, тогда как другие всегда только увеличивают затраты. Следующие факторы определяются качеством ***организации процесса производства*** и выполнения работ:

- степень, в которой процессы тестирования (модульного, компонентного, комплексного) регламентируют весь проект или присоединяются только в конце производства комплекса;
- четко определенные точки передачи управления между тестированием и остальными процессами производства модулей и компонентов;
- качество управления изменениями графиков разработки и тестирования, требований, архитектуры и тестирования модулей и компонентов;
- реалистичные и действующие графики и бюджеты производства и тестирования компонентов;
- своевременное и надежное устранение обнаруженных дефектов, при нарушении критериев качества тестирования компонентов.

Некоторые важные факторы вытекают из особенностей ***состава и квалификации коллектива тестирования***:

- навыки, опыт и отношение к работе в группе тестировщиков, особенно у тест-менеджеров и ключевых специалистов;
- взаимоотношения специалистов в группе тестирования компонентов;
- корпоративная культура и традиции коллектива специалистов;
- вдохновляемые и вдохновляющие менеджеры и технические лидеры, подготовленная группа управления, являющаяся организатором и контролером высокого уровни качества тестирования программных компонентов;
- понимание всеми участниками производства необходимости тщательного тестирования компонентов, подготовки версий, системного администрирования и документирования работ;

- стабильность производственного коллектива, отсутствие текучести специалистов;
- компетентная и своевременная поддержка тестовой среды;
- честность, обязательность, прозрачность открытых планов, доступность непосредственным исполнителям, менеджерам и заинтересованным лицам.

Несколько ***усложняющих производственных факторов***, которые всегда увеличивают сроки и затраты:

- большая сложность проекта, технологии производства, структуры предприятия или тестовой среды;
- большое число лиц, заинтересованных в тестировании, качестве компонентов и программном комплексе;
- большое количество малых групп, особенно если они разделены географически;
- необходимость обучать и вводить в производство растущую группу специалистов тестирования компонентов или всего проекта;
- необходимость осваивать или разрабатывать новые инструменты, методы и технологии как в тестировании, так и в производстве в целом;
- сложное расписание поступления протестированных модулей и компонентов, особенно для интеграции и комплексного тестирования.

Главное достоинство любого хорошего плана и прогноза в том, что он сбывается. Конечно, будет какое-то отклонение, ошибка на какой-то процент по времени и бюджету, но обычно половина оценок будет завышенной, а половина заниженной. Также ошибка не должна превышать 10 - 20%, и ни в коем случае оценка не должна отличаться от фактического результата в несколько раз. К сожалению, многие официальные графики и бюджеты проектов превышаются, часто на значительные величины. Это происходит по следующим ***основным причинам:***

Новые технологии – в некоторых случаях затраты недооцениваются, в которых они применяются, так как никто реально не знает, сколько времени нужно на выполнение работ, поскольку никогда ее не делали.

Производительность рабочей «команды» не достигает необходимого уровня эффективности или производительности из-за текучести кадров, усталости основных исполнителей, отсут-

ствия обучения, специалисты управления проектом не способны применять подходящие методы оценки затрат.

Отсутствие согласия – серьезные и непрерывные споры между ключевыми заинтересованными лицами мешают достижению согласия относительно того, что создается, в результате нельзя оценить затраты на тестирование компонентов, в постоянно меняющемся проекте, не может быть успешным выполнение всех работ производства.

Чрезмерные внешние ограничения – группы управления проектом создают графики и бюджеты в рамках цифр, диктуемых менеджерами, пользователями и другими людьми, не участвующими в процессах разработки комплекса программ и системы, их сопровождения или поставки, что является дополнительной проблемой.

Растянутые цели – группы управления производством объявляют цели, которые, как им заранее известно, не будут достигнуты, для того чтобы увеличить давление на непосредственных исполнителей и заставить их работать напряженнее.

Цель состоит в том, чтобы подготовить ***реалистичную оценку затрат***, а затем обсудить соответствующие изменения в рамках проекта и в тестовом покрытии тестирования компонентов в общем графике производства комплекса программ. Строгое следование правилам декомпозиции работ один из способов обеспечить полноту включения всей проектной группы, особенно опытных специалистов, в процесс оценки затрат. Для получения реалистичной оценки необходимо, чтобы она была основана на разумной продолжительности решения задач.

Группа тестировщиков должна начинать ***подготовку плана тестирования*** с получения от менеджера проекта или создания первичного шаблона плана тестирования модулей и компонентов, а затем уточнять план (см. рис. 2.12). Для планирования процессов ***программирования и тестирования компонентов*** целесообразно формировать график поставки компонентов и модулей для сборки функциональных групп программ. Компоненты в сложном комплексе программ обычно наиболее сильно взаимосвязаны в пределах решения крупных функциональных задач. Поэтому целесообразно создавать ***два уровня графиков планирования*** программирования и тестирования компонентов сложного комплекса программ: на уровне взаимодействия крупных функциональных задач и на уровне взаимодействия компонентов и модулей в составе функ-

циональных задач комплекса программ. При этом каждый компонент может быть ***вызывающим*** другие компоненты, и одновременно ***вызываемым*** из других компонентов и зависящим от результатов их функционирования. Эти связи определяют иерархическую схему взаимодействия между компонентами и модулями по управлению и по информации.

При программировании и тестировании ***при разработке снизу вверх*** (восходящая сборка) вызываемые компоненты предшествуют вызывающим, которые являются источниками информации для вызывающих, и могут использоваться как генераторы тестов для вызывающих компонентов. На графиках планов следует учитывать эти связи и распределять во времени поставку компонентов для сборки и комплексирования с учетом последовательности их времени разработки и иерархии связей. На графике планирования должны отражаться эти связи между моментами завершения разработки и тестирования вызываемого компонента и началом тестирования вызывающего компонента в составе фрагмента комплекса программ. Однако реальные процессы и последовательности программирования и автономного тестирования компонентов не всегда позволяют соблюдать рациональную логику последовательной разработки компонентов снизу вверх с учетом времени подготовки и взаимодействия в комплексе программ. Тогда для предварительного тестирования групп вызывающих компонентов сверху вниз, приходится разрабатывать временные имитаторы-заглушки для тестов, подменяющих вызываемые компоненты.

Планирование работ по тестированию должно учитывать ресурсы и работы, которые необходимо выполнить, чтобы своевременно ***подготовить тестовую среду*** для планомерного тестирования. Тестировщики каждого компонента должны определять требования к аппаратному, программному и сетевому обеспечению с целью создания и поддержки адекватных изменений тестовой среды. Нужно планировать работы по программированию, приобретению, установке и настройке компонентов, моделей или генераторов тестовой среды. Создание плана тестирования – ***итеративный процесс***, требующий обратной связи с различными участниками проекта и согласованности с определенными в нем процессами, стратегиями тестирования и сроками выполнения работ. С этим планом должен быть коррелирован и предшествовать план программирования и подготовки к тести-

рованию модулей и компонентов сложных функциональных задач. Тест-менеджер должен ***утвердить стратегию программирования компонентов, их тестирования*** и тестовые процедуры, которые должны быть подробно описаны в плане тестирования, и определять какие компоненты и модули, сценарии и тесты когда будут выполняться. Кроме того, предполагается, что руководитель проекта согласен с тем, что план тестирования компонентов и связанные с ним тестовые сценарии достаточно проверяют покрытие тестами требований, эталонов или сценариев использования комплекса программ и системы. Подробное изучение системных требований или сценариев применения системы вместе с тщательным определением параметров плана тестирования и требований к тестам необходимы для эффективного тестирования модулей и компонентов программного комплекса.

План тестирования должен определять и учитывать объем и длительность работ по тестированию каждого компонента. Обычно выстраивают ***структуру последовательности работ***, в которой на одном уровне определяются категории работ по тестированию компонентов, а на другом уровне – подробные описания работ. Структура детализации работ используется в сочетании с хронометражем для определения длительности выполнения этапов тестирования каждого компонента или модуля. Кроме того, план тестирования должен отражать ***оценки затрат*** на тестирование. Оценка затрат может определять число сотрудников группы тестирования компонентов в проекте в часах или в количестве специалистов, если для выполнения определенного объема работ выделяется конкретный срок. По возможности в план тестирования помещаются такие оценки затрат, как планируемое число тестовых процедур и сценариев.

Должны быть документированы ***квалификация и навыки сотрудников***, необходимых для проведения тестирования определенных компонентов. Состав группы тестирования, с требуемым качеством и навыками, может быть обозначен в требованиях к квалификации тестировщиков. Тест-менеджер должен оценить разницу между требуемой квалификацией и реальной подготовкой персонала, чтобы определить необходимые ***направления и объем обучения***. Планируемое обучение следует отразить в графике, чтобы соответствующие работы по тестированию компонента не предшествовали обучению. Также необходимо определить и документировать в плане роли

и ***ответственность сотрудников*** тестирования за качество определенного компонента с учетом особенности проекта.

Планирование тестирования компонентов должно быть сконцентрировано на определении и ***документировании требований к качеству результатов***, на создании документации для проведения тестирования, на планировании, необходимом для поддержки тестовой среды, и на разработке графика тестирования. С документированием требований к тестированию связана потребность в механизме хранения требований, используемом для управления. План тестирования должен иметь реальные ограничения по числу и квалификации сотрудников, по человеко-часам и по временному графику реализации каждого компонента. В плане тестирования также следует отражать требуемое качество, предварительные условия и допустимые риски результатов тестирования. Сюда включаются все события, действия или обстоятельства, которые могут помешать выполнению тестирования компонента в запланированный срок. С разработкой плана должно быть связано выделение функций, разработка которых имеет большое значение для успеха проекта, и функций, разработка которых связана с наибольшим риском. Определение наивысшего риска дает возможность группе тестирования ***сосредоточить усилия на функциях высокой значимости*** для пользователей и достоверности результатов тестирования.

Требования к тестам и результаты должны заноситься в ***базу данных и/или матрицу отслеживания реализации требований***. В базе данных или в матрице каждому требованию к тестам сопоставляется идентификационный номер реализующего его компонента системной архитектуры программного комплекса. Затем компонент архитектуры изучается вплоть до детализированных требований к программным модулям и до системных требований или сценариев использования комплекса программ и системы. После определения требований к тестам группа тестирования должна принять предварительное решение по ***методам тестирования***, которые наилучшим образом будут проверять каждое требование к компоненту и модулю.

Матрица отслеживания требований к тестам представляет собой продукт, получаемый, возможно, автоматизировано при помощи инструмента управления требованиями. Она позволяет проследить реализацию требований и сценарии использования комплекса программ, а также покрытие требований тестовыми про-

цедурами. Матрица может принимать одну из нескольких форм, основанных на различных видах отображений. Матрица отслеживания требований должна определять каждое требование, которое проверяется группой тестирования, а также метод его верификации. Важно то, что матрица отображает тестовые процедуры на системные требования или сценарии использования комплекса программ, и помогает убедиться тест-менеджерам в том, что они проверенны при тестировании компонентов и успешно реализованы.

После определения комплекса тестовых процедур во время планирования тестирования и проектирования тестов они должны заноситься в базу данных управления требованиями и привязываться к соответствующим требованиям или сценариям использования компонентов и комплекса программ. ***Управление требованиями к тестам*** включает в себя хранение требований, отслеживание связей, оценку рисков при реализации требований к тестам, выстраивание последовательности требований к тестам и определение методов верификации тестов. Отслеживание связей предполагает отображение тестовых процедур на требования к тестам и выявленных дефектов на тестовые процедуры.

Необходимо, чтобы группа тестирования как можно раньше получила информацию о матрице отслеживания требований от менеджеров проекта или конечных пользователей. Это способствует достижению согласия по поводу методов верификации, обеспечивающих проверку или контроль каждого требования к компоненту и комплексу программ. Принятие этого решения особенно важно, поскольку методы верификации отличаются сложностью и затратами времени. ***Раннее получение информации по матрице от менеджера*** позволяет группе тестирования увеличить допустимое время реакции на возможные изменения. Поскольку матрица отслеживания требований определяет выполняемые тестовые процедуры, одобрение матрицы менеджером или руководителем проекта отражает удовлетворительную степень покрытия тестами требований или сценариев компонентов и комплекса программ. При проведении приемо-сдаточных испытаний заказчик и тестировщики анализируют матрицу, чтобы проверить достаточность покрытие тестами требований или сценариев комплекса программ.

В плане тестирования программных компонентов должны быть определены требования к аппаратному, сетевому и программному обеспечению, что позволит ***создать тестовую среду***, являющуюся

отражением среды применения программного комплекса, предназначенного для тестирования. Приобретение, установка и настройка различных компонентов тестовой среды должно тщательно планироваться. При составлении плана тестирования компонентов определяются требования к тестовым данным и средствам для их получения, генерации или разработки. План тестирования должен отражать механизм ***управления целостностью тестовой модели***, обновления тестовой базы данных для возможности поддержки регрессионного тестирования. Группа тестирования должна получать ясную картину подготовительных работ (строительных блоков и заглушек), необходимых для создания тестовых процедур.

К ключевым элементам планирования тестирования компонентов относится ***тестирование технологической документации на компоненты и модули*** (см. лекцию 2.2). Группа тестирования модулей должна исчерпывающе документировать планы их тестирования, а тестировщики комплекса обязаны изучить содержание этих планов. Эта группа должна получить одобрение плана тестирования у менеджера проекта и может быть у конечного пользователя или заказчика. Они должны утвердить стратегию тестирования и тестовые процедуры, которые подробно описаны в плане тестирования, и определить, какие и когда тесты будут выполняться. Кроме того, предполагается, что руководитель проекта и/или заказчик согласен с тем, что план тестирования и связанные с ним тестовые сценарии правильно проверяют ***удовлетворительное покрытие тестами*** требований или сценариев использования комплекса программ и/или системы.

Успешный и рентабельный план тестирования ***требует ясного видения целей*** и четкого понимания различных параметров тестирования компонентов и комплекса программ. Планирование тестирования – ***не единовременный акт***, а ***процесс***. План – это документ для проведения тестирования от начала до конца разработки комплекса, и необходимо, чтобы он оперативно отражал все изменения в компонентах и комплексе программ. Группа тестирования должна постоянно обращаться к плану в ходе выполнения тестирования каждого компонента. Представление работ по тестированию в графической форме, позволяет специалистам по тестированию оценивать границы и масштаб затрат на тестирование. Структуру плана тестирования компонентов и комплекса программ обычно представляют двумя способами. Один метод организации тестовых процедур ***на базе ар-***

хитектуры комплекса программ и системы разбивает тестовые процедуры по функциям и компонентам системной архитектуры, по логическому принципу приоритетов в ее иерархии. Второй метод архитектура тестирования ***на базе выбранных методов*** связывает тестовые процедуры компонентов и модулей с определенными методами и инструментарием тестирования.

Разработка тестов включает создание тестировщиками, сопровождаемых, многократно применяемых и надежных тестовых процедур, что может потребовать ***не меньше усилий, чем разработка программистами тестируемых текстов программных компонентов*** (см. лекцию 2.1). Чтобы добиться максимального эффекта от автоматизации тестирования, тестировщики должны вести ***разработку и верификацию тестов параллельно с созданием и верификацией программистами текстов программных компонентов*** (рис. 2.13).

Рис. 2.13

Группе тестирования требуется проводить корректировку и уточнение структуры разработки тестов, чтобы отразить приоритеты конкретного программного комплекса.

Анализ и контроль полноты покрытия тестами требований и эталонов при исполнении компонентов комплекса программ должен определять, какие требования не были протестированы и какие части структуры компонентов и программного комплекса не были исполнены при тестировании. Тестовые варианты, основанные на требованиях и эталонах, могут не полностью покрыть структуру компонентов и комплекса программ. Анализ покрытия версии программного комплекса тестовыми данными, основанными на требованиях, должен определить, насколько полно тестирование проверило реализацию всех требований, и показать потребность в дополнительных тестовых сценариях. Поэтому должен выполняться анализ полноты структурного покрытия и проводиться его верификация. При тестировании реализации требований к функциям и характеристикам программных модулей и компонентов полнота их покрытия тестами редко достигает 90% и хорошо, если составляет около 80%.

Подготовка графиков разработки и выполнения тестов для модулей и компонентов комплекса программ

Группа тестирования должна составлять ***план-график разработки и выполнения тестовых процедур*** по шкале времени реализации проекта, как средство определения временных последовательностей для разработки и выполнения различных тестов (графики Ганта или сетевые графики). План-график определяет зависимости между тестами и общие сценарии, которые будут использоваться при тестировании. Перед созданием полного набора тестовых процедур для программного комплекса, тест-менеджер должен провести ***анализ и установить связи между программами модулей и компонентов***. Для этого строится матрица связей, которая показывает взаимозависимость различных модулей и компонентов и их тестовых сценариев. Графическое представление помогает тестировщикам определить возможности многократного применения сценариев в различных комбинациях с использованием модификаций, что позволяет свести к минимуму объем работ по созданию и сопровождению версий тестовых сценариев. В ходе разработки тестовых процедур группа тестирования должна проводить ***конфигурационное управление и контроль***

для тестовых сценариев и тестовых данных, а также для каждой отдельной тестовой процедуры. Необходимо создавать базовые версии тестового архива при помощи инструментов конфигурационного управления (см. лекцию 2.7).

Группа тестирования должна составлять график разработки и выполнения тестовых процедур, чтобы определить время и ресурсы на эти работы. ***При подготовке графика***:

- должны назначаться сотрудники, ответственные за выполнение каждой из работ по тестированию определенных модулей, компонентов и комплекса программ;
- следует учитывать последовательность разработки тестов, их взаимозависимость и связь с процессами и созданием компонентов и функций комплекса программ;
- тестовые процедуры могут быть объединены в группы в соответствии с функциями компонентов и программного комплекса;
- разработка тестовых процедур и их применение должны быть спланированы таким образом, чтобы исключить дублирование работ;
- в структуре тестовых процедур следует учитывать и документировать их приоритеты и риски;

План-график должен ***определить, кто из сотрудников отвечает*** за выполнение конкретного вида работ и компонентов тестирования. Также необходимо описать ***последовательность*** выполнения процедур тестирования и их ***взаимозависимость***, поскольку при тестировании определенная функция зачастую не может быть выполнена, пока предыдущая функция не сформирует нужные данные. Необходимо планировать работы по адаптации внешней среды, тестированию подготовки обязательных отчетов в графике разработки и выполнения тестовых процедур. Тестировщики должны иметь ***возможность выполнять свою работу независимо друг от друга*** и использовать одни и те же данные или базу данных тестов. График разработки и выполнения тестовых процедур должен позволять так организовать действия, чтобы один тестировщик не мог независимо изменять данные, используемые другим тестировщиком. При составлении графика разработки и выполнения тестовых процедур необходимо учитывать ***приоритеты*** и ***риски***, присвоенные различным тестам. График тестирования должен уделять внимание проверке функций, наиболее значимых для программного продукта, и функций повышенного риска. Такие тесты должны выполняться в первую очередь,

и график обязан предусматривать достаточно времени для проверки этих функций и, в случае необходимости, для регрессионного тестирования.

Менеджер тестирования должен отслеживать изменения требований и тестов и соответственно ***обновлять график тестирования***. Также производятся изменения графика тестирования, если планируемая функциональность не реализуется в данной версии. Нужно документально фиксировать каждое изменение плана-графика с помощью систем отслеживания графиков. Оригинальный план-график и все последующие изменения его базовых версий должны проходить процедуру утверждения у полномочных руководителей. Формальное утверждение каждой базовой версии плана-графика тестирования позволяет поддерживать прогнозы проведения тестирования в соответствии с реальным ходом работ.

Менеджеры должны осуществлять текущий контроль за выполнением тестирования модулей, компонентов и комплекса программ, подготавливая как ***внутренние отчеты*** о развитии каждого процесса, так и ***внешние обобщенные отчеты*** для руководителя проекта и/или заказчика в соответствии с условиями договора. Все обнаруженные дефекты и результаты их устранения должны быть документально оформлены, а также в установленные сроки подтверждена полная реализация процессов и выполнение утвержденных планов тестирования. После создания всех запланированных программных компонентов и комплекса, менеджер должен определить степень их ***соответствия критериям*** качества, установленным в договоре.

Стандартами **ISO 16326** и **ISO 90003** рекомендуется в процессе ***планирования производства комплекса программ*** подготовить и утвердить содержание следующих крупных ***детализированных планов***:

- производства модулей и компонентов комплекса программ, которые должна определять их модель жизненного цикл;
- верификации и тестирования, которые определяют методы и средства, способные удовлетворить последовательные цели процессов устранения дефектов и контроля качества программного комплекса;
- обеспечения качества модулей и компонентов программного комплекса, определяющего методы и средства, при помощи которых будет гарантировано их требуемое качество;

- реализации процессов интеграции модулей и компонентов в версии программного комплекса;
- сопровождения и управления конфигурацией программного комплекса, который должен устанавливать методы и средства, при помощи которых будут удовлетворяться цели процесса управления изменениями и корректировками компонентов комплекса программ;
- тиражирования, адаптации и внедрения версий программного продукта для конкретных пользователей;
- документирования процессов и результатов производства и выпуска технологической и эксплуатационной документации.

Каждый из перечисленных планов должен учитывать ресурсы, необходимые для его реализации, ***распределение работ на этапы, компоненты и временной график их выполнения***. В таком плане могут присутствовать ссылки на планы других видов, по которым они разрабатываются отдельно от этого плана. В ***процессе составления графика*** вся совокупность работ, необходимых для реализации продукта, разбивается на отдельные этапы и компоненты и оценивается время и затраты ресурсов, требующееся для выполнения каждого этапа и компонента. Обычно программирование и тестирование модулей и компонентов планируются и выполняются параллельно. График работ должен предусматривать это и распределять производственные ресурсы между ними оптимальным образом. Нехватка ресурсов для выполнения какого-либо критического этапа – частая причина задержки выполнения всего проекта. При расчете длительности этапов производства менеджер должен учитывать, что выполнение любого этапа разработки компонентов обычно ***не обходится без проблем и задержек***. Разработчики компонентов могут допускать ошибки или задерживать свою работу, техника может выйти из строя, аппаратные или программные средства поддержки процесса производства могут поступить с опозданием. Если проект инновационный и технически сложный, это становится дополнительным фактором появления непредвиденных проблем и увеличения длительности реализации проекта по сравнению с первоначальными оценками.

Особый вид ресурсов – это ***команда специалистов***, привлеченная к выполнению проекта. Существует эмпирическое правило планирования: оценивать ***временные затраты*** на этапы и компоненты так, как будто ничего непредвиденного не может случиться, а затем значительно (на 30 – 50%) ***увеличивать эти оценки*** для учета и компенсации

возможных проблем. Прогнозируемые проблемы существенно зависят от типа и размера проекта, а также от квалификации и опыта членов команды разработчиков. График работ по проекту обычно представляется в виде набора диаграмм, отражающих разбиение производственных работ на этапы и компоненты, зависимости между работами и распределение специалистов и затрат по этапам, работам и компонентам.

Даже в том случае, если итерационные методы предусматривают и допускают постепенное ***уточнение и расширение требований к комплексу программ***, целесообразно назначить некоторую дату, после которой в проект не должны вноситься новые требования. Рекомендуется включать в график резервные интервалы времени из-за того, что невозможно учесть действие всех внешних факторов. Если при этом в графике не предусмотрено буферного времени, то все временные производственные интервалы наползают друг на друга и начинают перекрываться. Одним из вариантов предупреждения таких возможных неожиданностей и конфликтов является отведение на этапы и компоненты заведомо большего количества времени. Другой вариант заключается в создании ***буферных резервных периодов в производстве***, во время которых не планируется решение каких-либо конкретных задач.

Применение графиков для планирования производства компонентов и комплексов программ

График становится более детальным по мере продвижения проекта и проведения ревизий. Как только утверждаются требования и архитектура комплекса программ, могут определяться частные производственные задачи и компоненты. В это же время могут более детально рассчитываться трудозатраты на каждый компонент и их взаимодействие, а также кто, когда и над чем будет работать. Чаще всего для представления графиков работ используются ***диаграммы Ганта***. Графики Ганта ***– это методика планирования проектов***, которую можно использовать для достижения нескольких целей, включая календарное планирование производства, финансовое планирование, планирование использования специалистов и различных ресурсов. График представляет собой гистограмму, где каждая горизонтальная линейка обозначает отдельный вид или компонент производственной деятельности (рис. 2.14).

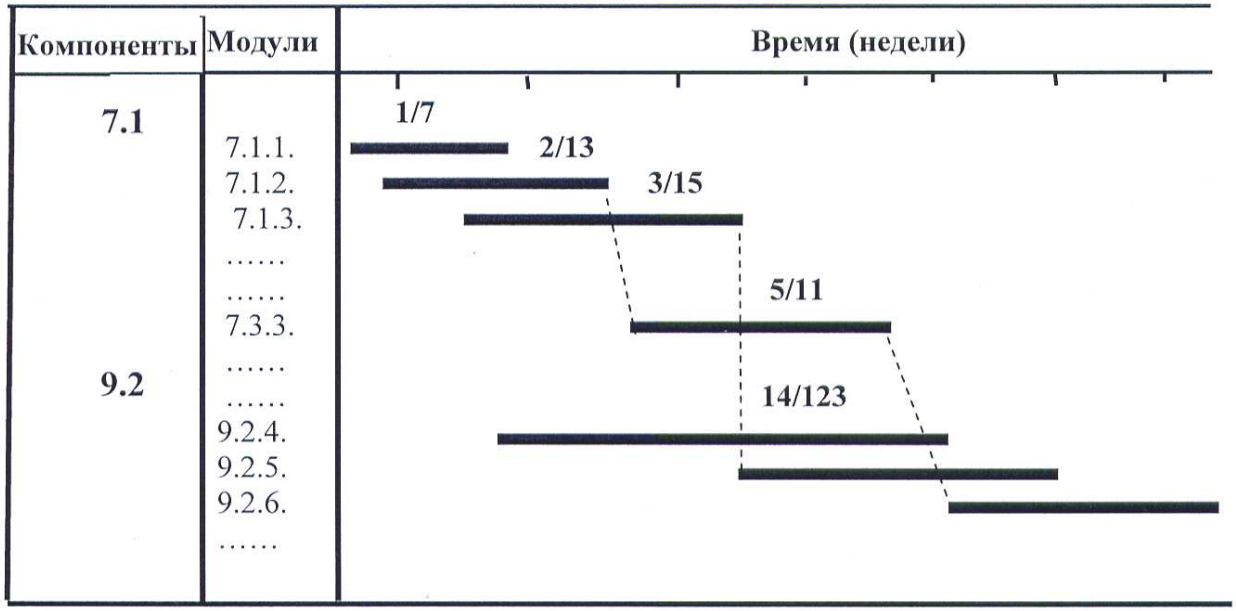


Рис. 2.14

Линейки изображаются относительно временной шкалы. Длина каждой линейки пропорциональна продолжительности времени, запланированного на выполнение определенной работы. Слева должны быть перечислены производственные действия, идентификаторы компонентов (например, 71 и 92) и модулей (7.1.1 … 9.2.6 и т.д.), а справа – указаны горизонтальные полоски, длина которых соответствует длительности выполнения каждого этапа производства и тестирования компонента в соответствии с временной шкалой. Визуально диаграмма Ганта представляет собой последовательность определенных действий, выполняемых в пределах жизненного цикла комплекса программ. Трудоемкость процессов и число необходимых специалистов может указываться числами над линейкой для каждого модуля или компонента (рис. 2.14 – в числителе – число специалистов, в знаменателе трудоемкость тестирования). Пунктирные линии отражают связи и зависимости начала некоторых работ только после завершения предшествующей работы, например, начало работы 9.2.5 только после завершения процесса 7.1.3.

Графики Ганта помогают выполнять ***календарное планирование производства***. Однако во время выполнения календарного планирования и его реализации могут быть определены новые виды деятельности, не предусмотренные во время первоначального укрупненного планирования. Тогда руководитель проекта должен вернуть-

ся назад и пересмотреть структуру декомпозиции работ и календарные планы для того, чтобы найти место и время вновь появившимся видам деятельности. При выполнении больших проектов, содержащих множество действий и другой отображаемой информации, диаграммы Ганта могут быть очень громоздкими. Это позволяет увидеть ***«общую картину» процессов производства***, однако ориентироваться в подобных графиках сложно. Поэтому полезно структурировать крупные графики на несколько более мелких по функциональным задачам или компонентам проекта, и на обобщающий сводный график производства всего программного комплекса без детализации процессов для компонентов.

График Ганта можно использовать для ***распределения ресурсов и планирования кадрового обеспечения***. Если необходимо спланировать по срокам работу конкретных специалистов, то можно воспользоваться графиком Ганта, на котором каждая полоса будет отражать одного из них. На таком графике ***специалисты являются ресурсами***, и на нем можно отражать их нагрузку во время производства. Можно пометить соответствующие части полос для обозначения количества времени, которое каждый специалист должен предположительно тратить на выполнение той или иной деятельности. Несмотря на то, что графики Ганта наглядно показывают каждый вид деятельности, они зачастую ***не отражают зависимостей между решаемыми задачами и компонентами***. Однако многие рабочие графики производства позволяют изображать связующие линии зависимостей между временем создания или тестирования модулей и компонентов (см. рис. 2.14 – пунктирные линии).

Независимо от того, как ***выявляются отклонения от плана***, руководитель – менеджер должен принимать решение об их устранении, Там, где производство ограничивается чисто человеческим трудом, обычно коррекция производится добавлением сотрудников или увеличением времени сверхурочной работы соответствующих специалистов. Однако в программировании недостаток чаще всего ощущается не в рабочей силе как таковой, а в ***интеллектуальных ресурсах специалистов***, поэтому с отклонениями от календарного плана здесь бороться сложнее. Формальное увеличение числа программистов не поможет «уложиться» в поставленные сроки, настолько же важно признать, что помочь в этом могут интеллектуальные способности только некоторых специалистов. Таким образом, приемлемым

вариантом может стать временный перевод подходящих специалистов на работу над проблемной частью проекта, либо наем специалистов – консультантов для устранения недостатков ресурсов.

Другой способ обеспечения уложиться в заданные сроки – подробное, критичное ***изучение требований к программному комплексу и исключение из них не самых актуальных***. Иногда, требования оказываются чересчур высокими, так как делаются попытки придать программному продукту дополнительные ***«украшения»***, не влияющий на него по существу. Избавление от ненужных требований состоит в «***чистке» эталонов – требований***. Важнейшим принципом планирования производства комплексов программ является ***упорядочивание приоритетов*** в порядке возрастания важности требований для заказчика и/или пользователей. Зачастую можно согласовать с заказчиком и принять решение об исключении определенных требований для того, чтобы уложиться в сроки. Следовательно, необходимо придерживаться принципа последовательного приращения функций и качества, не только на этапе анализа требований, но и во время планирования производства (в том числе календарного), реализации компонентов и комплекса программ.

Может случиться так, что после рассмотрения всех вариантов выход из цейтнота, подходящее решение найдено не будет. Тогда действием может быть ***признание неадекватности первоначальных планов и графиков***, и их пересмотр, исходя из нового понимания сложности поставленных задач, профессиональных способностей имеющихся специалистов и доступности ресурсов. Важно ***учитывать риски***, связанные с прогнозированием графика, и быть готовым к возможному пересмотру сроков. В менее тяжелых случаях принятие задержки результатов производства продукта может быть единственно правильным решением. Составляя, любой рабочий график, следует всегда учитывать ***принцип неопределенности***. Спланированный рабочий график – это предположение о будущих действиях разработчиков – программистов и тестировщиков при оптимальных условиях. Возможными последствиями неопределенностей можно управлять с помощью предусмотренных в графике ***временных резервов***, которые являются аналогами экономических и финансовых резервов, применяемых при построении бизнес-планов. Производство программного комплекса обычно должно быть окончено к определенной дате (что интересует руководителей проекта

и/или заказчика в первую очередь) ***с возможностью некоторой допустимой задержки***. Любое производство полно рисков, поэтому при объявлении предполагаемой даты завершения создания продукта, полезно указывать ***вероятность успеха***.

Для того чтобы ***эффективно управлять неопределенностью***, целесообразно использовать в графике начальные оценки длительности производственных этапов. В этом случае вероятность своевременного завершения каждого этапа или компонента будет составлять только 50%. Некоторую неопределенность можно внести в буферы завершающего действия, что при непосредственной работе над проектом позволит управлять неопределенностью ***сразу для всех производственных действий***. Вероятность своевременного завершения каждого этапа составляет только 50%, поэтому некоторые из них будут завершены позже установленного срока. Все разницы в сроках завершения этапов должны быть ***вычтены*** из буфера непредвиденных обстоятельств последнего этапа или всего производства в целом. Однако некоторые этапы могут быть завершены досрочно, поэтому выигрыш во времени также должен быть ***добавлен*** к буферу неопределенности проекта. Это означает, что размер буфера будет оставаться практически неизменным.

Для планирования и мониторинга планов производства, создано несколько ***универсальных технологических программных продуктов***. Они обеспечивают представление планов в различной графической форме – ***графиков Ганта, сетевых и других графиков***. Большинство из них не имеют проблемного ориентирования, и требуют перед применением подготовки состава, содержания и характеристик этапов, компонентов или работ, значений времени и трудоемкости, необходимых ресурсов и состава конкретных специалистов для наполнения графиков. В ***качестве примера***, достаточно простым и удобным может рассматриваться программный продукт **Microsoft Project** для представления, изменения и управления сложными планами производства в виде графиков Ганта. Основные функции пакета отражены ***четырьмя группами процедур***: задачи проекта; ресурсы проекта; отслеживание реализации и изменений проекта и отчеты. Эти группы позволяют создавать, манипулировать планами и конкретными работами с учетом их взаимосвязей, доступных ресурсов, локальных и общих целей планирования. Для крупных проектов воз-

можно выделение определенных этапов, компонентов и групп работ для построения из них локальных более детальных графиков Ганта.

Исходные данные о характеристиках компонентов проекта, необходимые для составления и описания плана, должны быть представлены в группе ***Задачи компонентов комплекса программ***: определение проекта; определение рабочего времени задач и компонентов; организация этапов и последовательности решения задач; установка крайних сроков и ограничений; определение допустимых рисков проекта от размещения задач.

Ресурсы проекта: выбор состава и квалификации специалистов; определение рабочих часов использования определенных ресурсов; назначение специалистов и оборудования на решение задач; публикация сводных данных о доступных ресурсах производства.

Отслеживание и изменение графика производства: создание и сохранение базового плана; организационная и техническая подготовка для отслеживания хода работ; контроль и изменение директивных сроков выполнения компонентов проекта; внесение изменений в компоненты, ресурсы и риски проекта; публикация данных о выполненных экономических планах, частей графика проекта.

Отчеты о результатах проекта и его графиках: отображение текущего состояния проекта; анализ и контроль критических задач экономики производства; контроль, сокращение или устранение рисков проекта; контроль выделения и использования времени и ресурсов; контроль и анализ экономических характеристик и ресурсов проекта; публикация сводных данных о реализации и графиках производства.

Графики Ганта позволяют динамически ***планировать производство программных комплексов***. Такое итерационное планирование целесообразно проводить, последовательно детализируя содержание производственных процессов с учетом размера, функций и характеристик компонентов конкретного комплекса программ в виде ***совокупности графиков***:

- этапов, производственных процессов и ресурсов всего жизненного цикла комплекса программ;
- каждой крупной функциональной задачи и взаимодействия ее модулей и компонентов в комплексе программ;
- программирования и тестирования модулей и компонентов каждой выделенной функциональной задачи.

Суммарные затраты ресурсов, уточняемые при их оценках по этапам, компонентам и по перечням работ, ***не должны выходить за пределы***, ограниченные договором. Если эти условия не выполняются, то ***итерационно следует***: пересматривать и изменять или ресурсы, или графики и распределение ресурсов по этапам и компонентам производства, или сокращать состав и содержание работ на производственных этапах компонентов. Во всех случаях следует контролировать обеспечение утвержденных требований к функциям и характеристикам программного продукта.

На основе значений длительности программирования и тестирования модулей и компонентов и взаимосвязи между ними может строиться ***сетевой график*** последовательности их программирования и тестирования. На этом графике должно быть видно, какие компоненты могут программироваться и тестироваться параллельно, а какие взаимозависимы и должны выполняться последовательно друг за другом. Некоторые компоненты могут начаться только тогда, когда будет ***получена*** контрольная отметка, которая может зависеть от разработки нескольких предшествующих компонентов. Они будут достигнуты только тогда, когда будет завершено тестирование компонента, в строке которого помещена соответствующая контрольная отметка.

Тестирование ряда определенных компонентов не может начаться, пока не выполнены все компоненты на всех путях, ведущих от начала тестирования компонентов комплекса к данному компоненту. Минимальное время выполнения всего тестирования комплекса можно рассчитать, просуммировав в сетевой диаграмме длительности тестирования ***на самом длинном маршруте*** от начала тестирования группы компонентов до его окончания (это так называемый ***критический путь)***. Таким образом, общая продолжительность тестирования компонентов проекта зависит от компонентов, находящихся на ***критическом пути.*** Любая задержка в завершении тестирования любого компонента на критическом пути приведет к задержке всего проекта.

Задержка в завершении тестирования компонентов, не входящих в критический путь, не влияет на продолжительность тестирования всего комплекса программ до тех пор, пока суммарная длительность тестирования этих компонентов (с учетом задержек на каком-нибудь пути) не превысит продолжительности работ на критическом маршруте. Сетевая диаграмма позволяет увидеть в зависимости длительности его тестирования, значимость того или иного компонента

для длительности реализации тестирования всего комплекса программ.

Первоначальный график работ неизбежно содержит ошибки или недоработки. По мере реализации проекта, предполагаемые длительности тестирования компонентов должны сравниваться с реальными сроками выполнения этих работ. Результаты сравнения должны использоваться в качестве основы для корректировки графика работ еще не реализованных компонентов проекта, в частности для того, чтобы попытаться уменьшить длительность критического пути. Важной частью работы менеджера проекта является ***оценка рисков***, которые могут повлиять на график работ или на качество создаваемого ***программного комплекса,*** и разработка мероприятий по предотвращению или сокращению рисков.

Лекция 2.4

ПОДГОТОВКА СРЕДСТВ ТЕСТИРОВАНИЯ КОМПЛЕКСОВ ПРОГРАММ НА СООТВЕТСТВИЕ ТРЕБОВАНИЯМ

Методы подготовки тестов для тестирования комплексов программ

Качество и тестирование программных модулей и компонентов, рассмотренные выше, являются базой для производства сложных комплексов программ высокого качества. Для окончательного определения достигнутого их качества необходимо тестирование и испытания комплексов в целом на соответствие исходным требованиям, утвержденным заказчиком и приемлемым потребителями. Эти требования отражают основную цель проекта, практическая реализация которой должна быть доказана достаточно полным тестированием с применением специфических методов и средств.

Как и при создании компонентов, ***разработка тестов для сложного комплекса в целом должна быть спланирована*** для того, чтобы проводимые в рамках тестирования работы привели к созданию наиболее эффективных тестов для целевой системы. Проектирование тестов и представление работ по тестированию в графической форме позволяет группе тестирования оценивать рациональные границы и масштаб тестирования. Исполняемые тесты могут быть ориентированы на разные уровни анализа ***функционирования*** комплекса и системы, обычно их называют ***структурными и/или динамическими***. Структурные тесты выявляют ошибки на основе внутреннего устройства, свойств комплекса программ и взаимодействия компонентов, а динамические тесты определяют ошибки на основе функционирования системы, которое имеет внешние проявления в выходных результатах. Ранние фазы тестирования обычно больше ориентируются на структурные тесты, а более поздние, как правило, концентрируются на динамических тестах соответствующих требованиям к комплексу.

Ручная пошаговая подготовка сценариев и содержания тестов не может обеспечить достаточно представительный их набор для тестирования крупных комплексов программ реального времени на соответствие требованиям к функциям и характеристикам качества. Для обеспечения высокого качества таких комплексов часто ориентируются на Альфа- и Бета-тестирование коллективами потенциальных пользователей или на тестирование при опытной эксплуатации. Однако эти методы приводят к длительным процессам устранения ошибок и дефектов, сохраняется неопределенность достигнутого качества и возможных рисков. Кроме того, такое тестирование может быть недопустимо для программных комплексов критических систем, где проявление случайных ошибок при эксплуатации может нанести большой ущерб.

Как правило, разработчиками моделей автоматизированной генерации тестов недооценивается сложность их разработки и пренебрегается необходимость детализации к ним требований. ***Требования и реализация совокупности тестов должны полностью отражать возможность проверки выполнения утвержденных требований к комплексу программ*** и соответственно должны быть принципиально аналогичными по сложности и трудоемкости разработке такого программного комплекса. Эти ***два представления комплексов программ*** отличаются только формой описания их содержания: функциональным (процессным) или событийным (сценариями и результатами исполнения). Генерация тестов особенно сложна (так же, как и требований к программному продукту) для крупных комплексов программ ***реального времени***. Выбор типов моделей генерации тестов зависит от глубины знаний об алгоритмах функционирования программ, характеристиках их качества и обобщенных параметрах необходимых результатов работы системы в целом. Кроме того, существенным для выбора типов моделей для генерации тестов может быть длительность расчета имитированных тестовых данных и обеспечение возможности проводить полную обработку результатов тестирования.

При наличии реальных планов и разумных предположений использование автоматизированных инструментальных средств и автоматизированных тестовых сценариев представляет собой ***способ снижеия временных и иных затрат на тестирование программного комплекса***. Эффективная Программа тестирования имеет свой

собственный ***жизненный цикл разработки***, в который входят планирование стратегий и целей, определение требований к тестам, анализ, проектирование и кодирование (рис. 2.15).

Подготовка средств тестирования сложных комплексов программ на соответствие требованиям должна включать:

- методы подготовки тестов для тестирования сложных комплексов программ:
  - классификацию методов подготовки тестов;
  - оценки методов автоматизации разработки тестов;
- требования к генерации динамических тестов внешней среды в реальном времени:
  - подготовки тестов на основе архитектуры системы;
  - задачи автоматизированного формирования динамических тестов внешней среды;
  - преимущества программной имитации динамических тестов;
- компоненты генераторов динамических тестов внешней среды в реальном времени:
  - исходные данные для генерации тестов внешней среды;
  - набор средств для формирования тестов внешней среды;
- обработку результатов динамического тестирования комплексов программ в реальном времени:
  - средства оперативной обработки результатов динамического тестирования;
  - средства оценки характеристик комплекса программ и системы;
- оценки эффективности динамической генерации тестов в реальном времени:
  - оценки затрат на программную имитацию тестов;
  - оценки экономической эффективности программной имитации тестов.

Рис. 2.15

По аналогии с процессом, которому следуют при разработке комплекса программ, требования к тестам необходимо определить прежде, чем тесты будут спроектированы. Требования к тестам должны быть ясно сформулированы и документированы, чтобы все участники проекта понимали, на чем основаны работы по тестированию комплекса.

Обычно на автоматизацию задачи уходит намного больше времени, чем на ее выполнение, поэтому для каждой задачи, которая может быть автоматизирована, целесообразно проводить тщательный ***анализ потенциального выигрыша от автоматизации***. Автоматизация тестирования обычно выражается в возможности формироваться и выполняться системы тестов без участия или при частичном участии человека. Процессы могут заключаться в автоматизированной работе с системой, предназначенной для тестирования, в загрузке данных для нее и в сравнении полученного результата с требуемым. Автоматизация тестирования должна позволять тестировщику спроектировать и разработать на основе исходных требований к комплексу полный комплект тестовых сценариев, а затем с небольшими затратами или вовсе без них ***повторять тестовые сценарии*** для обнаружения и устранения ошибок

Затраты, необходимые для автоматизации генерации тестов, могут быть не равны затратам на ручное написание тестов. Для создания автоматизированных тестов, удобных для развития и сопровождения, должна быть выстроена инфраструктура, позволяющая тестировщикам определять действия, данные и ожидаемые результаты в удобном формате. Наконец, автоматизация тестирования предполагает внесение изменений в процесс тестирования и освоенные навыки, необходимые группе тестирования, причем все это должны быть управляемо.

Если решено вложить средства в автоматизацию тестирования, следует проанализировать последствия этого решения в рамках выбранной стратегии. При предположении исполнять автоматизированные тесты в регрессионных испытаниях или для целей технического обслуживания системы может иметь смысл построить специализированный стационарный ***испытательный стенд*** генерации тестов, который будет находиться на рабочей площадке испытателей весь период, пока поддерживается и развивается программный комплекс. Однако такое решение влечет за собой существенное увеличение расходов на установку соответствующих аппаратных средств и на испытательный стенд.

Группа тестирования может при необходимости создавать специальные инструменты – ***генераторы динамических тестов***, которые на основе некоторого набора правил автоматически генерируют тесты. Эти правила можно получить из спецификаций ***требований к***

программному продукту из документации базы данных проекта; тестировщики могут разработать их вручную, чтобы привести в соответствие с требованиями к тестированию. При необходимости генераторы могут динамически создавать тестовые данные, например, для проведения нагрузочного или критических условий тестирования. Некоторые генераторы тестовых процедур могут обладать высокой степенью интеграции в процесс анализа и проектирования программных комплексов и позволять разработчикам тестировать функции в соответствии со спецификациями требований системной архитектуры. Другие генераторы тестовых процедур могут извлекать информацию о документированных требованиях из базы данных и создавать тестовые процедуры автоматически.

В результате тестирования сложных программных комплексов необходимо достоверно устанавливать ***степень их соответствия утвержденным требованиям к функциям и характеристикам качества***. Для этого они должны проходить тщательные динамические испытания в условиях их последующего применения. В реальных системах создание таких условий может быть очень дорого и опасно крупными рисками при проявлении не устраненных при тестировании дефектов и ошибок, что недопустимо. Альтернативой является создание и применение математических моделей на ЭВМ, ***динамически имитирующих реальную внешнюю среду*** для генерации тестов и функционирования тестируемых комплексов программ. Таким образом, формируется возможность выполнять тестирование, выявлять и устранять ошибки, а также ***достигать высокого уровня соответствия программного комплекса заданным требованиям***.

Критические просмотры и инспекции позволяют проводить первичную формальную оценку выполнения требований архитектуры, интерфейсов компонентов, структуры и содержания тестовых процедур и автоматизированных тестовых сценариев программных комплексов. ***Методика инспекций и критических просмотров*** один из важнейших элементов деятельности тестировщиков по созданию корректных программных комплексов. Они предполагают исчерпывающее обследование рабочих материалов компонентов и комплексов программ группой специалистов. Инспекции выявляют дефекты, отклонения от структурных стандартов разработки, спорные вопросы при создании тестовых процедур. Определение требований, являющихся доступными для тестирования и корректными, предотвращает возникновение ошибок, которые могли бы проявиться как дефекты

комплекса программ или системы на этапе разработки. Критические просмотры архитектуры проверяют ее ***согласованность с требованиями***, соответствие стандартам и применяемым методам проектирования, а также отсутствие в ней ошибок.

Инспекции и критические просмотры имеют ***несколько преимуществ***: они позволяют обнаруживать и устранять дефекты на ранних этапах цикла разработки и тестирования комплекса, предотвращать миграцию дефектов на поздние фазы создания продукта, повышать качество и производительность, снижать затраты и продолжительность жизненного цикла, а также сокращать объем работ по сопровождению. Результаты критических просмотров и инспекций должны соответствовать уровням качества, которые зафиксированы в ***стандартах разработки программных продуктов***, применяемых в данном проекте или в предприятии.

Деятельность по предотвращению дефектов следует концентрировать на устранении ошибок до начала кодирования модулей программ. ***Критические просмотры содержания и реализации требований*** заключаются в контроле требований верхнего уровня и позволяют убедиться в том, что компоненты и их взаимодействие соответствуют применяемым стандартам. Обычно просмотры предполагают использование списков контрольных вопросов для проверки того, что наиболее важные аспекты стандартов архитектуры и интерфейсов компонентов действительно и правильно применяются. ***При инспекциях*** это исследуют более детально, стимулируя составление разработчиками компонентов комментариев и обсуждение примерных тестовых процедур в группе разработки. Аналогично просмотры тестовых процедур выполняются на уровне контроля корректности реализации требований, тогда как инспекции являются более детальным исследованием тестовых процедур.

Метод структурного анализа, основанный на архитектуре, подразумевает обзор детализированного описания архитектуры комплекса. Метод анализа, основанный на архитектуре системы, позволяет сформулировать требования к тестам с использованием стратегии, проверяющей поведение внутренних компонентов системы, таких как управление, логика и потоки данных. Анализ, основанный на исследовании архитектуры, часто называют также структурным покрытием, что подчеркивает его связь со структурой комплекса программ.

Существуют три подхода к анализу ***требований к тестам на базе описания архитектуры***: покрытие компонентов, покрытие решений и модифицированное покрытие условий или решений. При использовании покрытия компонентов каждый компонент должен выполняться, по крайней мере, один раз. При покрытии решений каждая точка входа и выхода комплекса программ должны быть пройдены не менее одного раза, и хотя бы единожды рассматривается каждое решение в программе на предмет всех возможных исходов. Модифицированное покрытие условий и решений представляет собой критерий структурного покрытия, требующий выполнения каждого условия в составе решения с целью подтверждения того, что они независимо и корректно влияют на исход решения программного комплекса.

Если планируется проведение тестирования на уровне разработки системы, стоит проанализировать структурные требования к тестам, которые определяют необходимость проверки ***поведения системы в нештатных условиях***. Тестировщики должны сформулировать требования к тестам таким образом, чтобы предусмотреть выход за пределы массивов, запуск циклов на превышенное число итерации, использование неверных входных данных. Цель этих требований к тестам обнаружить ошибки, заставив программу работать в критических условиях.

Инспекции, проектирование и разработка тестов заранее, до завершения формулировки и утверждения требований к программному продукту, позволяет ***вступить тестировщикам в дискуссию с разработчиками требований*** о том, как система должна работать. В этот диалог включаются системщики и программисты, пользователи и аналитики, а также все другие заинтересованные лица проекта. По мере создания системы тестов, можно обсудить с заинтересованными лицами вопрос о том, что и как предполагается тестировать и какие результаты они ожидают получать.

Схема тестирования комплекса программ, основанная на архитектуре системы, связывает тестовые процедуры с аппаратным обеспечением и компонентами архитектуры комплекса. Логика этой модели строится на понимании того, что аппаратное обеспечение и компоненты архитектуры системы могут быть отслежены вплоть до спецификаций системных требований и требований к программному комплексу. Кроме того, формулировки требований к тестам можно отследить вплоть до компонентов архитектуры комплекса про-

грамм. Поэтому группа тестирования может обращаться к ***матрице отслеживания*** для выявления методов тестирования, которые соответствуют каждому компоненту архитектуры.

Модель Программы тестирования комплекса в графической форме должна отражать ее масштаб. Эта модель, как правило, показывает методы тестирования, необходимые для проведения динамического тестирования, и намечает стратегии статического тестирования. Определившись с моделью Программы тестирования, группа тестирования должна разработать архитектуру тестирования. Структуру архитектуры тестирования можно представить двумя способами. Один метод организации тестовых процедур, известный как тестирование ***на базе архитектуры системы***, разбивает тестовые процедуры по компонентам системной архитектуры по логическому принципу. Второй метод под названием тестирование ***на базе методов*** увязывает тестовые процедуры с различными методами тестирования, представленными в данной модели Программы тестирования.

Требования к генерации динамических тестов внешней среды в реальном времени

Для обеспечения высокого качества сложных комплексов программ реального времени необходимы соответствующие ***проблемно-ориентированные интегрированные системы автоматизации динамического тестирования***, способные достаточно полно заменить испытания программ с реальными объектами внешней среды (см. рис. 2.15). При этом высокая стоимость и риск испытаний с реальными объектами почти всегда оправдывают значительные затраты на такие интегрированные системы, если предстоят испытания критических программ с высокими требованиями к качеству функционирования программного комплексов, с длительным жизненным циклом и множеством развивающихся версий. При необходимости такие генераторы могут быстро создавать тестовые данные, например, для проведения нагрузочного тестирования. Некоторые генераторы динамических тестовых процедур могут обладать высокой степенью интеграции в процесс анализа и проектирования программных комплексов и позволять поэтапно тестировать их функции в соответствии со спецификациями требований и системной архитектурой. Другие генераторы могут извлекать информацию о документированных требовани-

ях из архивов программных комплексов и создавать тестовые процедуры автоматически.

Инструментальные средства автоматизации процессов динамического тестирования и испытаний крупных комплексов программ реального времени должны обеспечивать:

- определение и формирование динамических тестов – реализацию процесса тестирования разработчиком: ввод тестовых наборов; генерацию тестовых данных; ввод ожидаемых, эталонных результатов;
- выполнение участка тестируемого комплекса программ между контрольными точками, для которого средство тестирования может перехватить операторский ввод (клавиатуры, мыши и т.д.) и для которого вводимые данные могут быть отредактированы и включены в последующие тестовые сценарии;
- управление тестами и участком программы, для которого средство тестирования может автоматически выполнять тестовые наборы;
- анализ и обработку тестовых результатов – возможность средства тестирования автоматически анализировать части тестовые результаты: сравнение ожидаемых и реальных результатов; сравнение файлов; статистическую обработку результатов;
- анализ покрытия тестами исходных требований к программному продукту для обнаружения: функций, которые частично не были выполнены; процедур, которые не были вызваны; данных, к которым не были обращения;
- анализ производительности комплекса программ, когда он исполняется: загрузку центрального процессора; загрузку памяти; обращения к специфицированным элементам данных и/или сегментам кода; временные характеристики функционирования испытываемой программы;
- моделирование внешней среды – поддержку процесса тестирования с помощью модели динамической имитации данных из внешних для программного комплекса аппаратных компонентов системы.

Методы ***динамического тестирования с исполнением контролируемого комплекса программ***, в большей или меньшей степени, ориентированы на обнаружение ошибок определенных типов, преимущественно в структуре комплекса программ и реализуемых

маршрутах обработки информации. Методы тестирования потоков данных ориентируются на выявление ошибок в вычислительной части программ и в процессах преобразования различной информации. Такая ориентация позволяет упорядочивать последовательность приоритетного применения методов с целью устранения, прежде всего, ошибок в наибольшей степени отражающихся на корректности исполнения программ, а также сосредоточиваться на методах, позволяющих решать частные задачи достижения необходимого их качества и соответствия требованиям при минимальных затратах.

Характеристики динамического функционирования программных комплексов зависят не только от их внутренних свойств, но и ***от свойств внешней среды***, в которой они применяются (см. **ISO 12119**). Для сокращения неопределенностей и прямых ошибок при оценивании качества комплекса программ необходимо до начала испытаний определить основные параметры внешней среды и потоки информации, при которых он должен функционировать с требуемыми характеристиками при оценивании его качества и эксплуатации. Для этого заказчик и разработчик совместно должны ***структурировать, описать и согласовать модель внешней среды*** и ее параметры в среднем, типовом режиме применения, а также в наиболее вероятных и критических режимах, в которых должны обеспечиваться требуемые характеристики качества динамического функционирования программного комплекса. Такая ***модель должна отражать и фиксировать характеристики***:

- внешних динамических потоков информации, в том числе, их распределение по видам источников, характеристикам качества данных и возможности их дефектов;
- интенсивность и структуру типовых сообщений от оперативных пользователей и администраторов, и их необходимую квалификацию, отражающуюся на вероятности ошибок и качестве выдаваемой информации;
- возможных негативных и несанкционированных воздействий от внешней среды при применении программного комплекса;
- необходимые характеристики вычислительных средств, на которых предназначено функционировать комплексу программ с требуемым качеством.

При создании генераторов динамических тестов внешней среды применяется два принципиально различающихся подхода, которые

условно можно назвать интегральным и дифференциальным. При ***интегральном*** или эмпирико-статистическом подходе основой является формальное описание входной и выходной информации имитируемого динамического объекта, а также функциональной связи между данными на его входе и выходе. При этом структура объекта и процессы, реализующиеся при реальном функционировании его компонентов, не имеют значения и не моделируются. Исходные данные и характеристики для построения таких генераторов тестов получаются в натурных экспериментах или при исследовании более детальных – дифференциальных моделей.

Дифференциальные или имитационные модели генераторов динамических тестов базируются на описаниях внутренних процессов функционирования компонентов объекта моделирования, его структуры и взаимодействия составляющих. Результаты функционирования таких моделей определяются адекватностью знаний о компонентах и их характеристиках, а также об их взаимосвязях. Для этого необходимы достаточно подробные сведения о всех процессах функционирования компонентов объектов внешней среды, которые в свою очередь могут потребовать более глубокого моделирования их составляющих.

В отличие от натурного эксперимента моделирование внешней среды и динамических тестов на ЭВМ имеет большие возможности контроля**,** как исходных данных, так и всех промежуточных и выходных результатов функционирования испытываемого программного комплекса. В реальных системах ряд компонентов иногда оказывается недоступным для контроля их состояния, так как либо невозможно поместить измерители контролируемых сигналов в реальные подсистемы, подлежащие тестированию, либо это сопряжено с изменением характеристик самого анализируемого объекта. Преимуществом моделирования внешней среды на ЭВМ является также ***повторяемость результатов ее функционирования*** и возможность исследования большого количества вариантов и сценариев тестирования. В отличие от этого натурные эксперименты зачастую невозможно остановить на некоторой промежуточной фазе или повторить с абсолютно теми же исходными данными.

Программная имитация динамических тестов внешней среды на ЭВМ в реальном времени позволяет:

- проводить длительное непрерывное генерирование имитируемых данных для определения характеристик функционирования ком-

плекса программ в широком диапазоне изменения условий и параметров, что зачастую невозможно при использовании реальных объектов;

- расширять диапазоны характеристик имитируемых объектов за пределы реально существующих или доступных источников данных, а также генерировать динамические потоки информации, отражающие перспективные характеристики создаваемых информационных систем и объектов внешней среды;
- создавать тестовые данные, соответствующие критическим или опасным ситуациям функционирования объектов внешней среды, которые невозможно или рискованно реализовать при натурных экспериментах;
- обеспечивать высокую повторяемость имитируемых данных при заданных условиях их генерации и возможность прекращения или приостановки имитации на любых фазах моделирования внешней среды.

Компоненты генераторов динамических тестов внешней среды в реальном времени

Одними из наиболее сложных и дорогих имитаторов внешней среды, применяемых для испытаний крупных программных комплексов, являются модели: полета космических аппаратов; диспетчерских пунктов управления воздушным движением; объектов систем противовоздушной обороны; сложных административных или банковских систем и другие. Применяемые для этого моделирующие испытательные стенды (МИС) проблемно – ориентированы и размеры программ, моделирующих в них динамическую внешнюю среду, могут даже ***значительно превышать размеры*** соответствующих испытываемых программных продуктов. Для их реализации должны выделяться достаточно мощные универсальные ***моделирующие ЭВМ***. Кроме того, для автоматизации разработки программных комплексов могут использоваться отдельные ***технологические ЭВМ***, что в совокупности образует инструментальную базу для обеспечения всего ЖЦ сложных комплексов программ реального времени на ***объектных реализующих ЭВМ***.

Имитация конкретных динамических тестов с реальными характеристиками, адекватными объектам внешней среды, является основной частью типовых ***моделирующих стендов*** (см. рис. 2.15). В

соответствии с полной номенклатурой и реальными характеристиками таких объектов создаются их интегральные или дифференциальные модели. Выбор типов моделей зависит от глубины знаний об алгоритмах функционирования внешних объектов, характеристиках их компонентов и обобщенных параметрах работы объекта в целом. Кроме того, существенным для выбора типов моделей является длительность расчета имитированных данных и обеспечение возможности проводить полную имитацию динамической внешней среды на моделирующей ЭВМ ***в реальном времени*** с учетом затрат времени на обработку результатов.

Трудность адекватного моделирования некоторых динамических объектов внешней среды, особенно если при их функционировании активно участвует оператор-пользователь, не позволяет сосредоточить и полностью автоматизировать для крупных программных комплексов всю имитацию тестовых данных на моделирующих ЭВМ. Поэтому при реализации интегрированных проблемно-ориентированных МИС для испытаний качества функционирования сложных программных продуктов приходится использовать аналоги реальных объектов внешней среды для формирования части данных. Разумное ***сочетание части реальных объектов внешней среды и имитаторов на ЭВМ*** обеспечивает создание высокоэффективных МИС с комплексными моделями совокупностей объектов внешней среды, необходимых для динамических испытаний качества программных комплексов и систем в реальном времени. Такие стенды позволяют автоматическую генерацию тестов с помощью имитаторов на ЭВМ и аналогов реальной аппаратуры дополнять реальными данными от операторов пользователей, контролирующих и корректирующих функционирование систем обработки информации.

В схеме типового МИС можно выделить ряд базовых компонентов, назначение и функции которых представлены на рис. 2.16. Для каждого эксперимента при динамических испытаниях комплексов программ реального времени следует подготавливать план сценариев тестирования и ***обобщенные исходные данные.***

В моделирующей ЭВМ план и обобщенные исходные данные преобразуются в конкретные значения параметров для задания динамического функционирования каждого имитатора или реального объекта внешней среды.

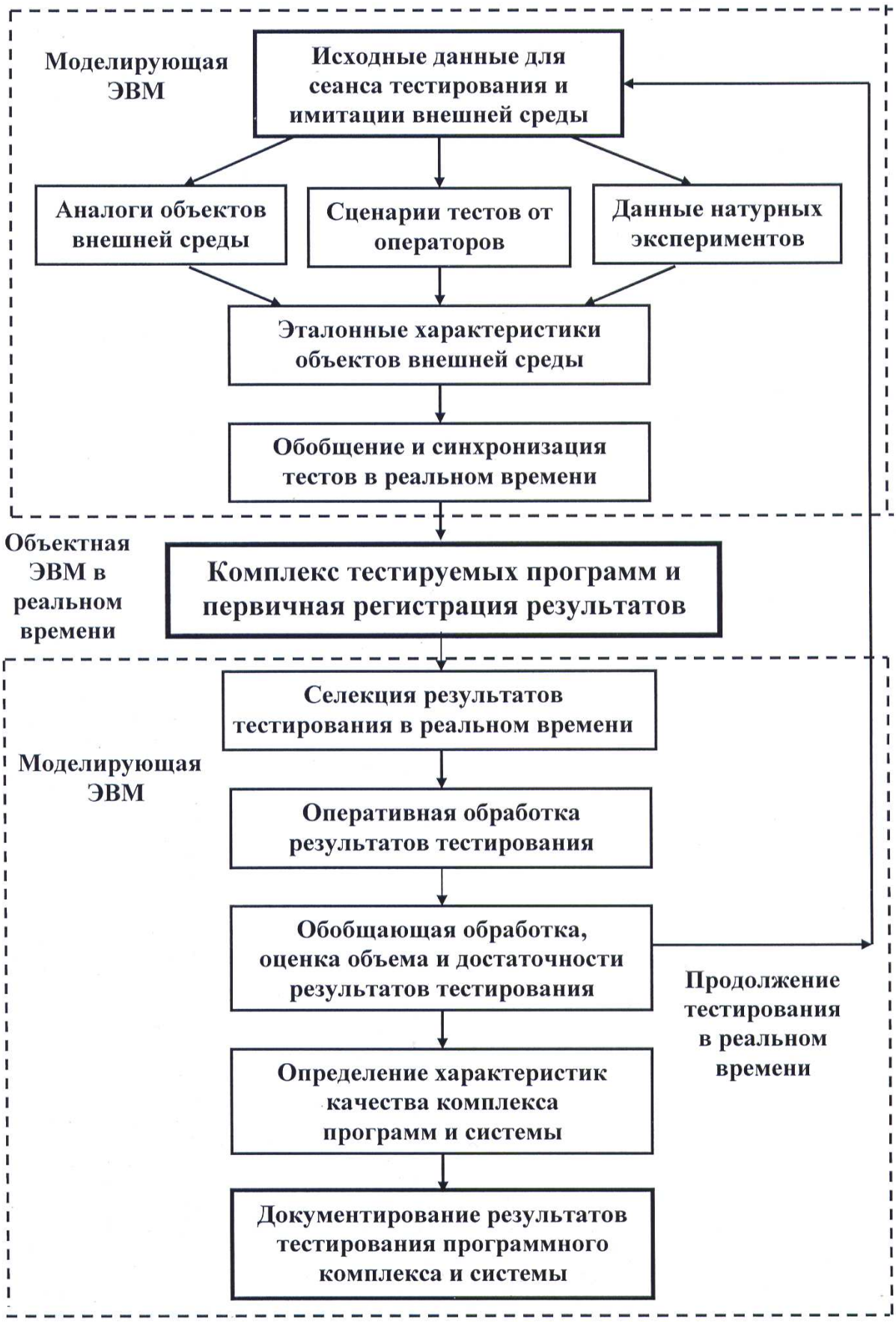

Моделирующая ЭВМ
Исходные данные для сеанса тестирования и имитации внешней среды
Аналоги объектов внешней среды
Сценарии тестов от операторов
Данные натурных экспериментов
Эталонные характеристики объектов внешней среды
Обобщение и синхронизация тестов в реальном времени
Объектная ЭВМ в реальном времени
Комплекс тестируемых программ и первичная регистрация результатов
Моделирующая ЭВМ
Селекция результатов тестирования в реальном времени
Оперативная обработка результатов тестирования
Обобщающая обработка, оценка объема и достаточности результатов тестирования
Продолжение тестирования в реальном времени
Определение характеристик качества комплекса программ и системы
Документирование результатов тестирования программного комплекса и системы

Рис. 2.16

Эти данные вводятся и преобразуются на моделирующей ЭВМ вне реального масштаба времени и подготавливают старт сеанса функционирования стенда и испытываемых программ в реальном времени. После этого начинают генерироваться тестовые данные.

Аналоги системы и аппаратуры внешней среды используются преимущественно для генерации динамических тестов, представляющих коррелированные логические переменные, которые трудно описать и смоделировать на ЭВМ. Кроме того, они позволяют проверить и аттестовать некоторые программные имитаторы внешней среды, которые впоследствии играют основную роль при испытаниях. В ряде случаев такие аналоги аппаратуры не могут отразить все особенности объектов внешней среды, и имитаторы на ЭВМ остаются единственными источниками соответствующей части данных для проверки качества программного комплекса.

Данные с рабочих мест операторов-пользователей должны отражать реальные характеристики динамических воздействий на тестируемый программный комплекс с учетом особенностей и квалификации человека, которому предстоит использовать испытываемые программы в реальной системе обработки информации. На эту часть МИС кроме первичных исходных данных от моделирующей ЭВМ могут вводиться данные обработки ряда тестов испытываемой системой. В результате через человека и его характеристики замыкается контур обратной связи ручного и автоматизированного управления динамическими объектами внешней среды. Такое же замыкание контура автоматизированного управления возможно в аналогах и аппаратных имитаторах реальных объектов.

Данные натурных экспериментов с объектами внешней среды могут подготавливаться заранее вне сеансов динамических испытаний программного комплекса, например, при отладке аппаратной части системы обработки информации. Эти данные отражают характеристики и динамику функционирования объектов, которые трудно или опасно подключать для непосредственного взаимодействия с недостаточно проверенными программами. Кроме того, такие данные могут использоваться для аттестации адекватности имитаторов некоторых объектов внешней среды. Они могут быть полезны в тех случаях, когда создание определенных условий динамического функционирования объектов внешней среды очень дорого или опасно и может быть выполнено только в исключительных случаях. Однако данные натурных экспериментов не всегда удается адекватно описать

условиями и обобщенными характеристиками их поведения. Наличие ряда случайных неконтролируемых факторов усложняет картину исходных данных и может затруднять сопоставление результатов натурных экспериментов с полученными при программной имитации тестов.

При динамическом тестировании в ряде случаев необходимо иметь ***эталонные характеристики данных***, поступающих на испытываемый программный комплекс или систему. При работе с реальными объектами зачастую приходится создавать специальные измерительные комплексы, которые определяют, регистрируют и подготавливают к обработке все необходимые характеристики в процессе реального функционирования этих объектов (например, координаты движения самолетов при испытаниях систем УВД). Такие измерения проводятся при автономном функционировании внешних объектов или при их взаимодействии с комплексом программ в реальном времени. Имитация эталонных характеристик объектов внешней среды может служить для определения качества функционирования программного комплекса в идеальных условиях – при отсутствии искажений исходных данных, ошибок в измерениях их параметров, сбоев и предумышленных отказов аппаратуры. Проверка при таких исходных данных позволяет оценить характеристики дефектов и ошибок результатов, обусловленные недостаточным качеством комплекса программ.

Синхронизация и обобщение тестовых данных предназначены для упорядочения динамических тестов от источников различных типов, в соответствии с реальным временем их поступления на комплекс программ, и для распределения между моделирующей и объектной ЭВМ. В результате формируются потоки тестовых данных каждого реального объекта внешней среды, которые вводятся в объектную ЭВМ в соответствии с логикой функционирования системы обработки информации через соответствующие устройства сопряжения с моделирующей ЭВМ.

Повторяемость сеансов испытаний при автоматической имитации динамических тестов обеспечивается фиксированием всех исходных данных и применением программного формирования псевдослучайных чисел. При надежной работе аналогов реальных объектов и моделирующей ЭВМ, в принципе, можно добиться ***почти абсолютной повторяемости*** весьма длительных экспериментов и сце-

нариев тестирования. Некоторая не идентичность результатов при повторных экспериментах может быть обусловлена сбоями и частичными отказами аппаратуры. Труднее обеспечивать повторяемость сценариев динамических испытаний, в которых активно участвует оператор-пользователь. В этом случае необходимо регистрировать и сохранять действия оператора в зависимости от времени, а затем повторять их в соответствии с записанным сценарием. При необходимости временная диаграмма может соблюдаться с точностью около 0,5-1 с, однако ошибки в действиях оператора и вводимых им параметрах могут отличаться в каждом сценарии тестирования. Вследствие этого повторяемость тестов реализуется только статистически.

Приведенные выше рекомендации по функциям и применению МИС ориентированы на создание крупных программных комплексов, их динамическое тестирование и испытания, в основном, до передачи в регулярную эксплуатацию. После приемки заказчиком или приобретения пользователями в процессе функционирования и применения программного комплекса должно ***обеспечиваться их регулярное тестирование*** и оценка текущего качества. Для этого в ***составе программного комплекса необходимы средства, обеспечивающие:***

- генерацию динамических тестовых сценариев или хранения тестов для контроля работоспособности, сохранности и целостности программного комплекса при функционировании и применении;
- оперативный контроль и обнаружение дефектов исполнения программ и обработки данных при использовании программного комплекса по прямому назначению;
- реализацию процедур предварительного анализа выявленных дефектов и оперативное восстановление вычислительного процесса, программ и данных (рестарт) после обнаружения аномалий динамического функционирования программного комплекса;
- мониторинг, накопление и хранение данных о выявленных дефектах, сбоях и отказах в процессе исполнения комплекса программ и обработки данных.

Средства генерации динамических тестов и имитации внешней среды в составе поставляемого программного комплекса предназначены для оперативной подготовки исходных данных при проверке различных режимов функционирования в процессе применения программного комплекса и при диагностике проявившихся дефектов. Минимальный состав средств генерации тестов должен передаваться пользователям для контроля использования рабочих версий

в реальном времени и входить в комплект поставки каждой пользовательской версии программного комплекса. Для размещения таких средств мониторинга и контроля качества функционирования крупных комплексов программ необходимы ресурсы памяти, а также дополнительная производительность ЭВМ. Более глубокие испытания функционирования версий и локализации ошибок следует проводить на базе комплекса средств ***имитации внешней среды высшего уровня в МИС на моделирующей ЭВМ***, которые используются специалистами для испытаний и/или сертификации системы. Часть этих средств имитации может применяться как средства нижнего уровня (пользовательские) на объектной ЭВМ для диагностики и обеспечения полного повторения ситуаций, при которых пользователями могут быть обнаружены динамические дефекты функционирования.

Важной функцией испытательных стендов является их использование в качестве ***тренажеров для операторов-пользователей***. Так как качество функционирования комплексов программ может существенно зависеть от характеристик конкретного человека, участвующего в эксплуатации и обработке информации, то необходимо измерять эти характеристики. Необходимо также иметь возможность их улучшать до уровня, обеспечивающего выполнение ***заданных требований к программному продукту***. Поэтому в средства тестирования и испытаний органически должно входить обеспечение процессов динамической тренировки и измерения характеристик реального функционирования и реакции операторов, а также использование МИС для обучения и регулярной подготовки операторов - пользователей в процессе тиражирования и эксплуатации программного продукта.

Важнейшее значение для определения качества программных комлексов имеет ***адекватность имитаторов динамических тестов***, которая зависит от степени учета второстепенных факторов, характеризующих функционирование реальных объектов или источников информации, при создании их моделей. Точность моделей на ЭВМ, прежде всего, определяется алгоритмами, на которых они базируются, и полнотой учета в них всех особенностей моделируемых объектов. Кроме того, на адекватность имитации влияет уровень дефектов и ошибок в программах имитации. Каждый, не учитываемый в имитаторе элемент или фактор моделируемой системы, необходимо оценивать путем сопоставления частных имитируемых данных с результатами аналитических исследований или с данными, полученны-

ми на реальных системах, и определять его возможное влияние на полную требуемую точность модели и генерируемых динамических тестов с учетом других составляющих, отражающихся на достоверности имитации. Перечисленные факторы, влияющие на достоверность генерации тестов в МИС, взаимозависимы, и повышение достоверности имитации за счет одного из факторов при ограниченных ресурсах приводит, как правило, к снижению достоверности вследствие влияния остальных. Поэтому важной задачей при создании имитационных моделей является достижение наибольшей суммарной достоверности имитации и определения значений характеристик качества функционирования программного комплекса, при сбалансированном влиянии каждого из факторов. Поэтому при создании сложных генераторов тестов необходимо достигать, по возможности, равное влияние отмеченных факторов на суммарную достоверность оценки качества программного комплекса при квалификационном тестировании и испытаниях.

Обработка результатов динамического тестирования комплексов программ в реальном времени

Регистрация и обработка характеристик динамических тестовых данных должна обеспечивать их контроль на соответствие заданным требованиям к программному комплексу, обобщенным характеристикам каждого объекта внешней среды и исходным данным сеанса испытаний. Так проводится процесс испытаний по конечным результатам функционирования комплекса программ, выдаваемым внешним абонентам и для определения интегральных характеристик качества и их соответствия требованиям к программному комплексу. Однако для диагностики и локализации отказов, дефектов и ошибок, а также для оценки некоторых частных характеристик качества могут быть необходимы промежуточные данные исполнения программ на объектной ЭВМ. Для регистрации промежуточных данных следует иметь возможность разорвать процесс естественного исполнения комплекса программ на любом заданном компоненте, операторе или при обращении на запись или чтение к конкретным данным в объектной ЭВМ.

Разрыв динамического исполнения программ для анализа промежуточных данных может производиться методом вставок специальных контрольных программ при подготовке комплекса программ к

конкретному сеансу испытаний. Такие вставки при их небольшом числе почти не искажают реальный масштаб времени. Они обычно размещаются в завершающей части отдельных функциональных групп или компонентов комплекса программ, позволяют контролировать и локализовать причины отказов и дефекты с точностью до достаточно крупных участков комплекса программ. В точках контроля и разрыва естественного процесса исполнения комплекса программ обычно размещаются только операторы ухода на специализированную группу программ регистрации и оперативной обработки промежуточных данных в объектной ЭВМ. Далее эти данные либо накапливаются и предварительно обрабатываются в объектной ЭВМ, либо оперативно передаются в моделирующую ЭВМ для более глубокой обработки.

Селекция результатов динамических испытаний может основываться на стратегии контроля функционирования программного комплекса ***снизу вверх***, т.е. от анализа исполнения отдельных компонентов программы и далее до стохастических результатов функционирования всего комплекса в динамике реального времени. При этом регистрируется избыточное количество данных, из которых затем отбирается минимум, необходимый для анализа. Может использоваться стратегия ***сверху вниз***, т.е. упорядоченное, иерархическое выделение в первую очередь обобщенных результатов функционирования комплекса программ с последующим уточнением регистрируемых и анализируемых результатов вплоть до детального контроля исполнения отмеченных программных компонентов. В этом случае регистрируются только те данные, которые необходимы для анализа в конкретном сеансе динамического тестирования. При обеих стратегиях необходимо иметь возможность управлять объемом и видом выделяемой и регистрируемой информации тестирования в зависимости от целей испытаний. Данные, получаемые и выделяемые в процессе испытаний качества комплекса программ, целесообразно делить на следующие ***группы***:

- данные, характеризующие исходную тестовую информацию и выходные результаты динамического тестирования;
- маршруты исполнения программных компонентов и их операторов при некоторых фиксированных тестовых данных;

- аномальные события, сбои, отказы и данные, характеризующиеся отклонением результатов динамического тестирования от требований за допустимые пределы и ограничения;
- характеристики динамического использования различных ресурсов объектной ЭВМ.

Регистрация промежуточных данных обычно соответствует некоторым достаточно завершенным этапам испытаний функционирования программного комплекса. Вызовы регистрирующих программ должны подчиняться определенной системе контроля динамического функционирования программ при исходной гипотезе, что ***некоторые ошибки и дефекты в программах и данных могут проявиться на любой стадии тестирования.*** Однако количество вызовов регистрирующих программ и контроль промежуточных результатов, требующих нарушения целостности исполнения функциональных программ, следует ограничивать, учитывая допустимые расходы ресурсов времени на их реализацию. Так как основная задача регистрации при тестировании в реальном времени состоит в обнаружении и локализации ошибок и причин отказов с точностью до функциональной группы программ или компонента, то более точное определение места дефекта приходится переносить на тестирование компонентов и модулей по детерминированным сценариям вне реального времени.

Так как испытания современных крупных систем обработки информации и/или управления позволяют получать такое большое количество контрольных данных, что достаточно полный их анализ представляет трудную методическую и техническую задачу, ***обработку динамических результатов целесообразно осуществлять иерархически и дифференцировано*** (см. рис. 2.16). При избытке контролируемых величин снижается общее быстродействие имитаторов и комплекса программ в результате затрат времени на контроль и регистрацию. При переходе к массовым экспериментам испытаний функций и реализации их качества, приходится значительно сокращать количество анализируемых параметров и по возможности представлять их в обобщенном виде. В каждом конкретном случае необходимо стремиться к компромиссу между полнотой регистрации промежуточных данных тестирования и удобством ***анализа обобщенных результатов на соответствие требованиям***.

Обработка результатов испытаний программных комплексов реального времени может быть разделена на две достаточно автономные части: оперативную и обобщающую. ***Оперативная обработка***

результатов динамического тестирования должна производиться по упрощенным алгоритмам с большой пропускной способностью, обеспечивающим сохранение реального масштаба времени для всего испытываемого комплекса программ. Основная часть оперативной обработки результатов связана с замыканием контура обратной связи для имитации динамики функционирования управляемых объектов внешней среды. Оперативно следует производить селекцию некоторых результатов тестирования и их предварительную обработку для значительного сокращения объема сохраняемых результатов.

В оперативную обработку целесообразно включать расчет части интегральных данных динамического тестирования, позволяющих контролировать текущий процесс обработки информации испытываемым комплексом. Желательно выделять, регистрировать и отображать критические значения параметров или ситуации, угрожающие надежности и безопасности функционирования. Объем таких оперативно отображаемых данных должен быть максимально сокращенным и в то же время достаточным для анализа критических ситуаций, отражающихся на качестве динамического функционирования программного комплекса. Эти данные должны позволять специалистам, ведущим испытания, фиксировать условия, при которых проявляются дефекты в функционировании программ, с учетом того, что автоматическая регистрация всегда имеет пробелы в составе фиксируемых параметров.

Обобщающая обработка накопленных результатов испытаний может производиться вне реального времени после завершения одного или серии испытаний. Основная задача при этом состоит в расчете различных интегральных характеристик качества функционирования программного продукта и ***соответствия их заданным требованиям***.

Зарегистрированные и обработанные результаты испытаний должны использоваться для установления соответствия полученных характеристик качества ***заданным требованиям***. При выявлении их отклонения от требований технического задания заказчика, спецификаций требований или декларируемых в документации, должны разрабатываться корректировки программ для устранения несоответствия. Для этого все этапы тестирования и испытаний программных продуктов должны быть поддержаны системой конфигурационного управления версиями программных компонентов и базой данных до-

кументирования тестов, результатов испытаний и выполненных корректировок программ (см. лекцию 2.7 и стандарты **ISO 10007** и **ISO 15846**). Средства накопления сообщений об отказах, ошибках, предложениях на изменения, выполненных корректировках и оцененных характеристиках качества версий являются основой для конфигурационного управления развитием и совершенствованием комплекса программ.

Затраты на имитацию внешней среды и на генерацию динамических тестов для оценивания качества программных комплексов могут быть одной из существенных составляющих при их создании. В ряде случаев они соизмеримы с затратами на создание основных функций комплексов программ, что определяется принципиальным соответствием ***сложности необходимых наборов тестов*** и тестового покрытия программ, и ***сложности функций***, реализуемых испытываемым комплексом программ (см. лекцию 1.1). Создание представительных совокупностей динамических тестов возможно путем использования реальных объектов внешней среды или с помощью программных имитаторов, адекватных этим объектам по результатам функционирования и генерируемой информации. При этом возникает проблема – ***какой метод и когда выгодней*** по затратам на генерацию тестов и по обеспечению необходимой степени покрытия тестами испытываемых комплексов программ.

Имитаторы тестов могут быть необходимы не только для оценивания достигнутых характеристик качества комплексов программ, но также для их комплексной отладки, квалификационного тестирования, испытаний и при создания версий. Поэтому затраты на программные имитаторы и их экономическую эффективность целесообразно рассматривать в проекте с учетом всего комплекса задач, которые они способны и должны решать в жизненном цикле программного комплекса. Анализ эффективности программной имитации внешней среды при разработке и определении качества целесообразно разделять ***на две части***: оценка факторов, определяющих эффективность средств имитации тестов, и оценка экономического выигрыша при моделировании внешней среды на ЭВМ по сравнению с натурными экспериментами в реальных системах.

Факторы, определяющие эффективность программной имитации внешней среды на ЭВМ при разработке крупных комплексов программ, могут оцениваться в основном по их воздействию на качество создаваемых систем. Это влияние трудно непосредственно изме-

рить, однако качественный анализ показывает, что автоматизированная имитация динамических тестов может значительно изменять не только достигаемые характеристики качества разрабатываемого программного комплекса, но также трудоемкость и длительность его создания. Программная имитация внешней среды на ЭВМ может обеспечивать широкие наборы тестов и достаточно полные тестовые покрытия комплексов и компонентов при испытаниях, в том числе а пределами характеристик реально существующих или доступных источников тестов, а также соответствующие критическим или опасным ситуациям динамического функционирования объектов внешней среды. Для каждого параметра, отражающего внешнюю среду, отношение диапазона или числа тестов, возможных при программной имитации на ЭВМ по сравнению с натурными экспериментами, может служить оценкой величины, возрастания достоверности характеристик качества программных комплексов.

Некоторые значения тестов не только трудно создать при натурных экспериментах, но они являются маловероятными в реальных условиях. Однако такие, даже маловероятные, ситуации и значения тестов могут быть ***критическими и/или особо важными*** для функционирования всей системы, для которой разрабатывается программный комплекс. Выбор и имитация подобных ситуаций позволяют отрабатывать и оценивать качество в критических маловероятных ситуациях, которые невозможно или опасно создавать на реальных объектах, но без их выполнения некоторые продукты не допустимо эксплуатировать в критических системах управления и обработки информации.

Экономическую эффективность программной имитации внешней среды на ЭВМ по сравнению с натурными экспериментами целесообразно оценивать при одинаковых объемах динамических тестовых данных для испытаний и определения качества. Показателем экономической эффективности имитации может служить ***соотношение затрат*** на проведение натурных экспериментов и затрат на программную имитацию той же совокупности тестовых и эталонных данных. Затраты ресурсов на натурные эксперименты для генерации тестов при проведении разработки, испытаний и определения качества пропорциональны реальному времени функционирования проверяемого программного комплекса и затратам на применение привлекаемых средств реальной внешней среды. Они включают стоимость

эксплуатации реального объекта, создающего динамические тесты в единицу времени (например, затраты на функционирование административной системы, прокатного стана или системы управления воздушным движением и всех управляемых ею объектов). Таким образом, затраты на натурные эксперименты для оценивания характеристик комплексов программ определяются использованием всей реальной внешней среды, в которой предстоит в дальнейшем функционировать программам, а также затратами на средства измерения характеристик этой среды и проверяемого программного комплекса в процессе разработки, испытаний и определения качества.

Затраты на программную имитацию динамических тестовых данных определяются ресурсами необходимыми на проектирование и эксплуатацию сложных комплексов программ для этих целей. Имитационные стенды практически всегда являются уникальными. В ряде случаев эти комплексы программ могут иметь объем порядка 10^6 строк текста и должны создаваться с применением современных технологических систем. Затраты на эксплуатацию программ имитации в основном определяются длительностью проведения динамического тестирования, испытаний и/или измерения характеристик качества. Значения этого времени соответствуют реальному времени генерации тестовых данных и тестирования программ. Затраты на эксплуатацию ЭВМ, используемую в моделирующем имитационном стенде, включают: первичные затраты на закупку и установку оборудования, необходимого для имитации тестовых данных, стоимость имитирующей ЭВМ и устройств сопряжения имитационного стенда с ЭВМ, на которой функционируют тестируемые программы.

Даже приближенные оценки при системном анализе соотношения этих затрат в большинстве случаев показывают ***высокую рентабельность программных имитаторов внешней среды***, особенно для квалификационного тестирования и оценивания характеристик качества крупных программных комплексов реального времени. Например, при тестировании системы для управления воздушным движением, применение имитационных стендов, по крайней мере, на порядок снижает затраты по сравнению с натурными экспериментами и использованием реальных объектов (самолетов), а для управления космическими аппаратами или атомными электростанциями это соотношение может быть значительно больше (~ 10 – 100). При создании и определении качества административных систем с полной загрузкой, имитация способна заменить сложную организацию функционирова-

ния по определенной программе большого коллектива операторов банка, налоговой инспекции или таможенного органа.

Примером сложного испытательного стенда и моделей внешней среды для динамического тестирования и проверки ***на соответствие требованиям*** к функциям и характеристикам комплексов программ может рассматриваться система ***управления полетами воздушных судов и диспетчерских систем в центрах управления воздушным движением***. Для комплексной отладки, тестирования, испытаний и сертификации программных продуктов управления воздушным движением (УВД) проводится имитация в реальном времени всей информации, поступающей из внешней среды. Источниками информации для центров УВД являются радиолокационные станции, летный состав на борту воздушных судов, диспетчеры управления воздушным движением и исходные планы полетов.

Лекция 2.5

ТЕСТИРОВАНИЕ ПРОГРАММНЫХ КОМПЛЕКСОВ НА СООТВЕТСТВИЕ ТРЕБОВАНИЯМ К ХАРАКТЕРИСТИКАМ И ДОКУМЕНТАМ

Тестирование надежности функционирования программных комплексов

В составе требований к системам и программным комплексам, функционирующим в реальном времени, особое значение имеют ***динамические функции и характеристики*** (см. лекцию 1.3). Для обеспечения их высокого качества непригодны отдельные сценарии и процедуры тестов, а необходимо создавать ***потоки динамических тестов в реальном времени***, адекватные соответствующим данным при функционировании внешней среды систем и/или пользователей (рис. 2.17). Эти потоки тестов должны обеспечивать динамическую проверку комплексов программ на соответствие требованиям, выявление дефектов и ошибок при реализации их функций и характеристик в реальном времени. Основная особенность такого тестирования состоит в необходимости создания динамической среды функционирования программного комплекса, максимально приближенной к реальной, при его практическом применении.

Задача состоит в определении соответствия требованиям, функций и характеристик программного комплекса при различной интенсивности потоков тестов, адекватных нормальным условиям применения программного комплекса, а также критическим по составу и интенсивности, для выявления предельных условий его работоспособности. Такие ***условия тестирования отражаются на интегральных характеристиках***, на снижении надежности и/или безопасности, а также на повышении рисков применения программного комплекса. Для комплексов программ реального времени ***особое значение могут иметь*** причины и методы уменьшения ***рисков надежности и производительности***, вследствие дефектов и ошибок, а

также при формировании и реализации требований к этим характеристикам.

Тестирование программных комплексов на соответствие требованиям к характеристикам и документам должно включать:

- тестирование надежности функционирования программных комплексов:
  - экспериментальные методы оценивания надежности программных комплексов в штатном режиме;
  - форсированные испытания для оценивания надежности программных комплексов;
  - повышение надежности комплексов программ, путем оперативного контроля и рестарта;
- тестирование функциональной безопасности программных комплексов:
  - оценивание эффективности тестирования функциональной безопасности программных комплексов;
  - удостоверение достигнутой функциональной безопасности систем с программными комплексами;
- тестирование характеристик производительности и использования ресурсов ЭВМ программными комплексами:
  - нарушения временного баланс между длительностью решения задач и производительностью ЭВМ;
  - оценивания предельной пропускной способности системы с программным комплексом;
  - определение качества динамического функционирования программного комплекса при перегрузке;
- тестирование документации на соответствие требованиям к программным комплексам:
  - определение характеристик аудитории пользователей программного комплекса;
  - подготовка требований и плана эксплуатационной документации программного комплекса;
  - тестирование реализации документов на соответствие требованиям к функциям и характеристикам программного комплекса;
  - тестирование эксплуатационной документации на практичность.

Рис. 2.17

Эти взаимосвязанные характеристики качества программных комплексов зависят от одних и тех же свойств воздействий из внешней среды, требуют совместного анализа и методов для выявления и устранения дефектов. Локализация и устранение таких динамических дефектов обычно осуществляется вне реального времени, путем применения детерминированных сценариев и тестовых процедур, а иногда за счет изменения требований заказчика.

Оценивание надежности программных комплексов включает измерение количественных характеристик: завершенности, устойчивости к дефектам, восстанавливаемости и доступности-готовности (см. лекцию 1.3). При этом предполагается, что в контракте, техническом задании или спецификации требований зафиксированы и утверждены заказчиком определенные значения этих характеристик и их приоритеты. Измерения проводятся при функционировании готового программного комплекса для сопоставления с заданными требованиями и для оценивания рисков соответствия этим требованиям. Тестирование для оценки надежности комплекса программ должно проводиться в тестовом окружении, которое максимально приближено к реальным условиям применения системы. Входные параметры тестов следует задавать на основе вероятностного распределения соответствующих характеристик или их наборов при эксплуатации программного комплекса.

Значения надежности коррелированны с характеристикой корректность, однако можно достигать высокую надежность функционирования комплекса программ при относительно невысокой их корректности за счет сокращения времени восстановления при отказах. Кроме того, надежность можно оценивать косвенно в процессе разработки по прогнозируемой плотности обнаружения скрытых дефектов и ошибок, а также по плотности выявляемых и устраняемых ошибок выходных результатов при тестировании динамического функционирования комплекса программ. Степень покрытия тестами структуры функциональных компонентов и комплекса программ в целом может служить ***ориентиром для прогнозирования их потенциальной надежности***. Распределение реальных длительностей и эффективности восстановления при ограниченных ресурсах для функционирования программ, может рассматриваться как дополнительная составляющая при оценивании надежности.

Для прямых, количественных измерений надежности необходимы инструментальные средства, встроенные в операционную систему

или в соответствующие компоненты комплекса программ. Эти средства должны в динамике реального функционирования программ, автоматически селектировать и регистрировать аномальные ситуации, дефекты и искажения вычислительного процесса, программ и данных, выявляемые аппаратным, программно-алгоритмическим контролем или пользователями. Накопление и систематизация проявлений дефектов при исполнении программ позволяет оценивать основные показатели надежности, помогает определять причины сбоев и отказов и подготавливать данные для повышения надежности программных комплексов. Регулярная регистрация и обобщение таких данных способствует устранению ситуаций, негативно влияющих на функциональную пригодность и другие важные динамические характеристики.

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

При испытаниях надежности в первую очередь обнаруживаются ***отказы – потери работоспособности.*** Однако в большинстве случаев первоначально остается неизвестной причина происшедшего отказа. Для выявления фактора, вызвавшего отказ (первичной ошибки или дефекта) и устранения его причины, необходимо, прежде всего, определить, каким компонентом системы стимулирован данный отказ. Для диагностики и устранения случайных редких отказов должна быть организована ***служба их регистрации*** с максимально полным фиксированием характеристик ситуаций, при которых проявился каждый. Для выявления ***тенденции изменения показателей надеж-***

ности, их зарегистрированные значения необходимо связывать во времени с моментами корректировки программ. Анализируя корреляцию между значениями надежности и процессом изменения программ, можно выявлять некоторые корректировки, которые содержат ошибки и снижают надежность.

При заключительных приемо-сдаточных и сертификационных испытаниях ***для достоверного определения надежности*** организуются многочасовые и многосуточные прогоны динамического функционирования комплекса программ в реальной и/или имитированной внешней среде в условиях широкого варьирования исходных данных с акцентом на стрессовые ситуации, стимулирующие проявления ***угроз надежности***. Такие прогоны позволяют измерять достигнутые характеристики надежности и определять степень их соответствия требованиям технического задания, а также закреплять их в технических условиях и документации на программный комплекс.

Если интенсивное тестирование программ в течение достаточно длительного времени не приводит к обнаружению дефектов или ошибок, то у специалистов, ведущих испытания, создается ощущение бесполезности дальнейшего тестирования программного продукта, и он передается на эксплуатацию. Экспериментальное исследование характеристик сложных ПС позволило оценить ***темп обнаружения дефектов – риски, при котором крупные программные продукты передаются на регулярную эксплуатацию***: 0,002–0,005 дефектов в день на человека, т.е. специалисты по испытаниям или все пользователи в совокупности выявляют только около одной ошибки или дефекта каждые два – три месяца использования ПС. Интенсивность обнаружения ошибок ниже 0,001 ошибок в день на человека, т.е. меньше одной ошибки в год на трех-четырех специалистов, непосредственно выполняющих динамическое квалификационное тестирование и/или эксплуатацию комплекса программ, по-видимому, может служить ***эталоном высокой надежности*** обработки информации. Если динамическое функционирование программного продукта происходит непрерывно, то эти показатели соответствуют высокой наработке на обнаружение дефекта или отказа порядка 5 – 10 тысяч часов и коэффициенту готовности выше 0,99.

Форсированные (стрессовые) испытания для оценивания надежности (а также функциональной безопасности и рисков) программных комплексов значительно отличаются от традиционных ме-

тодов испытаний аппаратуры. Основными факторами, влияющими на надежность, являются исходные данные и их взаимодействие с дефектами и ошибками программ или сбоями в аппаратуре ЭВМ. Поэтому форсирование испытаний надежности осуществляется повышением интенсивности искажений исходных данных и расширением варьирования их значений, а также специальным увеличением интенсивности потоков информации и загрузки программ на ЭВМ выше нормальной.

При форсированных испытаниях целесообразно выделять следующие ***режимы тестирования***:

- полное искажение, предельные и критические значения ключевых параметров тестов каждого типа внешней информации и воздействий пользователей;
- предельные и критические сочетания значений различных взаимодействующих параметров тестов при эксплуатации программного комплекса;
- предельно большие и малые интенсивности суммарного потока и каждого типа внешней информации;
- умышленные нарушения пользователями определенных положений инструкций и рекомендаций эксплутационной документации на программный комплекс.

Особым видом форсированных испытаний является целенаправленное тестирование эффективности средств оперативного контроля и восстановления программ, данных и вычислительного процесса ***для оценивания восстанавливаемости***. При таких испытаниях основная задача состоит в оценивании качества динамического функционирования средств автоматического повышения надежности, и в измерении характеристик восстанавливаемости. Для этого имитируются запланированные условия функционирования программ, при которых в наибольшей степени стимулируется срабатывание средств программного рестарта и оперативного, автоматического восстановления работоспособности. Следует особо отметить трудности достижения и регистрации надежности программ, характеризующейся наработкой на отказ >>100 ч. При такой надежности резко возрастают сложность обнаружения возникающих отказов и диагностирования их причин.

В любых ситуациях функционирования сложных комплексов программ, прежде всего, должны исключаться катастрофические последствия и длительные отказы или в максимальной степени смяг-

чаться их негативное влияние на результаты, выдаваемые пользователю. Неизбежность дефектов и ошибок в сложных комплексах программ, искажений исходных данных и аппаратурных сбоев приводит к необходимости регулярного контроля процесса исполнения комплекса программ, и сохранности данных. Предвидеть заранее все ситуации исполнения программ и протестировать при них крупные программные комплексы оказывается невозможным из-за их огромного количества, поэтому применяются методы, которые направлены на ***оперативное обнаружение последствий*** дефектов и аномального функционирования программ, а также на автоматическое восстановление (рестарт) нормального вычислительного процесса и искаженных текстов программ и данных. Для обеспечения высокой надежности функционирования необходимо максимально быстро обнаруживать аномалии, достаточно точно классифицировать тип уже имеющихся и возможных последствий искажений, а также осуществлять мероприятия, обеспечивающие быстрое восстановление нормального функционирования программного комплекса и системы.

Введение средств контроля функционирования и помехозащиты в программы позволяет ***скомпенсировать влияние на надежность неполной корректности программных комплексов***, а также снизить негативные воздействия внешних возмущений различных типов. Восстановление работоспособности при этом желательно производить настолько быстро, чтобы отказовую ситуацию можно было свести до уровня кратковременного сбоя. Визуализация отклонений от нормы при динамическом функционировании позволяет пользователям контролировать аномалии в процессе обработки данных и в особых случаях оперативно корректировать реакцию системы защиты на выявленные искажения.

Особенности тестирования функциональной безопасности программных комплексов

Функциональная безопасность программных комплексов и систем зависит от отказовых ситуаций, негативно отражающихся на работоспособности и реализации их основных функций, причинами которых могут быть дефекты и аномалии в аппаратуре, комплексах программ, данных или вычислительных процессах (см. рис. 2.17). При этом катастрофически, критически или существенно искажается процесс функционирования программных комплексов и/или систем,

что наносит значительный ущерб при их применении. Основными источниками отказовых ситуаций могут быть некорректные исходные требования, сбои и отказы в аппаратуре, дефекты или ошибки в программах и данных функциональных задач, проявляющиеся при их динамическом исполнении в соответствии с назначением. При таких воздействиях, внешняя, функциональная работоспособность систем может разрушаться не полностью, однако невозможно полноценное выполнение заданных функций и ***требований к программному комплексу***.

Понятия, методы тестирование и характеристики функциональной безопасности программных комплексов и систем близки к аналогичным для надежности. Поэтому способы оценки и испытаний функциональной безопасности могут базироваться на методах тестирования, определения и обеспечения надежности функционирования комплексов программ. При более или менее одинаковых источниках угроз и их проявлениях эти понятия можно разделить ***по величине негативных последствий*** и ущерба при возникновении отказовых ситуаций. Чем сложнее системы и чем выше к ним требования безопасности, тем неопределеннее функции и характеристики тестирования требований для обеспечения их безопасности. Неопределенности начинаются с требований заказчиков, которые при формулировке технического задания и спецификаций ***не полностью формализуют*** и принципиально не могут обеспечить достоверное содержание всего адекватного набора характеристик и значений требований безопасности, которые должны быть при завершении проекта и предъявлении конечного программного продукта заказчику. Эти требования итерационно формируются, детализируются и уточняются по согласованию между всеми участниками проекта.

Всегда не полностью, с необходимой детализацией определены и описаны все характеристики, особенности функционирования и безопасности объектов внешней среды. Эти характеристики в той или иной степени обычно находятся под воздействием управляемой системы. Квалификация и субъективные свойства потребителей и пользователей изменяются по мере освоения функциональных возможностей системы и ее работоспособности, что увеличивает неопределенность ее реальной безопасности. Различия свойств персонала, применяющего систему, дополнительно увеличивают неопределенность значений безопасности и трудности ее прогнозирования при тестиро-

вании с учетом множества ***субъективных факторов различных специалистов, участвующих в эксплуатации***.

При анализе характеристик функциональной безопасности целесообразно выделять и учитывать особенности ***двух классов систем и их программных продуктов***. Первый класс составляют системы, имеющие встроенные комплексы программ жесткого регламента реального времени, автоматизировано управляющие внешними объектами или процессами. Время необходимой реакции на отказовые ситуации таких систем обычно исчисляется секундами или долями секунды, и процессы восстановления работоспособности должны проходить за это время, в достаточной степени автоматизировано (бортовые системы в авиации, на транспорте, в некоторых средствах вооружения, системы управления атомными электростанциями). Системы второго класса, применяются для управления процессами и обработки деловой информации из внешней среды, в которых активно участвуют специалисты-операторы (банковские, административные, штабные военные системы). Допустимое время реакции на опасные отказы в этих системах может составлять десятки секунд и минуты, и операции по восстановлению работоспособности частично могут быть доверены специалистам-администраторам по обеспечению функциональной безопасности.

Эти факторы влияют на ***неопределенность критериев, методов и оценивания значений эффективности тестирования функциональной безопасности*** конкретных программных комплексов и систем. Существующие технологии тестирования способствуют повышению функциональной безопасности, снижению потенциального ущерба и рисков, однако практически всегда остается открытым вопрос, насколько применяемые методы оправдывают затраты на реализацию требований заказчика.

Роль негативных воздействий и их разрушительные последствия быстро возрастают в связи с ростом сложности разработки и применения современных систем на базе ЭВМ и ответственности решаемых ими задач. Одновременно возрастает сложность внешней и операционной среды, в которой функционируют комплексы программ и ***ответственность функций систем, связанных с безопасностью***. Объективное повышение сложности функций, реализуемых программными комплексами в современных системах, непосредственно приводит к увеличению их объема и трудоемкости создания. Соответственно росту сложности программ возрастает относительное и

абсолютное количество выявляемых и остающихся в них дефектов и ошибок, что ***отражается на снижении потенциальной безопасности их функционирования***.

Достижение требуемой функциональной безопасности систем, содержащих программные комплексы реального времени, решается путем использования ***современных регламентированных технологических процессов динамического тестирования***, подобных применяемым при обеспечении надежности. Они должны быть поддержаны группой международных стандартов, определяющих состав и процессы выполнение требований к заданной функциональной безопасности систем и комплексов программ. Для систематической, координированной борьбы с угрозами безопасности программ необходимы исследования факторов, влияющих на функциональную безопасность со стороны случайных дефектов и ошибок, существующих и потенциально ***возможных в конкретных системах и комплексах программ***. Это позволяет целенаправленно разрабатывать и применять методы и средства обеспечения функциональной безопасности критических программных комплексов различного назначения при реально достижимом снижении уровня дефектов и разработки. Проблема в значительной степени решается посредством применения современных методов, инструментальных средств и стандартов, поддерживающих системный анализ, технологию проектирования, разработки, тестирования и сопровождения систем, и их программных комплексов.

Тестирование характеристик производительности и использования ресурсов ЭВМ программными комплексами

Оценивание ресурсной эффективности состоит в измерении количественных характеристик: временной эффективности и используемости динамических ресурсов ЭВМ комплексом программ (см. рис. 2.17). При этом предполагается, что в контракте, техническом задании и спецификации требований зафиксированы и утверждены требуемые значения этих характеристик и их приоритетов.

Цель тестирования производительности – продемонстрировать заказчику, что система функционирует в соответствии с требованиями, содержащимися в спецификациях на производительность и приемлемого времени отклика при обработке заданного количества

транзакций. При тестировании производительности должны применяться промышленные нагрузки, что позволяет предсказать поведение программного комплекса и системы при реальной эксплуатации. Средства, обеспечивающее тестирование производительности, должны позволять оценивать влияние перегрузок.

Оценивание перегрузок – это процесс тестирования работоспособности вычислительных машин при обработке большого потока данных с целью выяснения того, когда и где программный комплекс выйдет из строя под высокой нагрузкой. Эти ***допустимые пределы должны быть определены в системных требованиях*** к программному комплексу, где также должна определяться реакция системы на перегрузки. Данный вид тестирования необходим для систем, работающих с максимальной спроектированной нагрузкой, для проверки того, что они динамически функционируют в соответствии с требованиями.

Адекватное проведение динамического ***тестирования программного комплекса на перегрузки***, при использовании ручных методов подготовки тестов – дорого, трудоемко, неточно и занимает много времени. В распределенных системах требуется большое количество пользователей и рабочих станций для осуществления внешней среды, обеспечивающей процесс динамического тестирования. Выделение значительных ресурсов для тестирования требует больших затрат, и трудно организовать совместную работу необходимого числа пользователей и машин. Необходимы средства автоматизированного тестирования, которые включают имитаторы нагрузки и позволяют тестировщикам динамически имитировать сотни виртуальных пользователей или объектов, одновременно работающих с целевым программным продуктом.

Для ***измерения характеристик временной эффективности*** необходимы инструментальные средства, встроенные в операционную систему или в соответствующий комплекс программ. Эти средства должны в динамике реального функционирования программного комплекса регистрировать:

- загрузку вычислительной системы функционирующими программами;
- значения интенсивности потоков данных для обработки от конкретных внешних абонентов;

- длительность исполнения типовых заданий при реализации конкретных функций;
- характеристики функционирования устройств ввода/вывода;
- время ожидания результатов (отклика) на функциональные задания пользователей или системы;
- заполнение памяти обмена с внешними абонентами в различных режимах применения программного комплекса.

Значения этих характеристик зависят не только от свойств и функций комплекса программ, но также от особенностей архитектуры и операционной системы ЭВМ. Регулярная регистрация и обобщение таких данных позволяет выявлять ситуации, ***негативно влияющие*** на функциональную пригодность, надежность и другие важные характеристики программного комплекса. Существует особый вид тестов для проверки удовлетворения специфических требований, предъявляемых к ***параметрам производительности***. При этом может производиться динамическое тестирование с целью достижения реальных (достижимых) ***возможностей*** по производительности, и функционирования программ с повышением нагрузки, вплоть до достижения ***запланированных характеристик требований*** и далее, с отслеживанием их поведения на всем протяжении повышенной загрузки системы.

При излишне высокой интенсивности поступления исходных данных может ***нарушаться временной баланс*** между длительностью решения требуемой совокупности задач программным комплексом в реальном масштабе времени, и производительностью ЭВМ при решении этих задач. Также возможно нарушение баланса между имеющейся в ЭВМ памятью и памятью, необходимой для хранения всей поступившей и обрабатываемой информации. Для выявления подобных ситуаций и определения характеристик программного комплекса в условиях недостаточности ресурсов ЭВМ проводятся испытания при высокой, но допустимой, в соответствие с требованиями, интенсивности поступления исходных данных.

При использовании комплексом программ производительности и памяти реализующей ЭВМ менее чем на 50%, разработчик может практически не учитывать эти ограничения и сопутствующие риски. Закономерным является стремление разработчиков программ применять, особенно для систем реального времени, встроенные объектные ***ЭВМ с предельным использованием их технических характери-***

стик. Опыт создания программ реального времени позволяет утверждать, что практически невозможно использовать производительность объектной ЭВМ более чем на 95%, и почти всегда целесообразно ограничиваться на уровне 80 – 90%, так как иначе, затраты на разработку программного комплекса могут значительно увеличиться. Подобная зависимость обусловлена сложностью оптимального распределения в динамике ограниченных ресурсов ЭВМ (особенно производительности) по многим функциональным задачам, необходимостью проектирования комплекса программ с учетом этих ограничений и неоднократными переделками программных компонентов для того, чтобы соблюсти ресурсные ограничения.

Для оценивания характеристик использования производительности при тестировании крупных программных продуктов ***должны быть измерены***:

- реальные значения интенсивностей поступающих исходных данных и заданий на вызов функциональных программ, а также распределения вероятностей этих интенсивностей для различных источников и типов заданий;
- длительности автономного решения отдельно каждой функциональной задачи, обрабатывающей исходные данные или включаемой внешними заданиями, а также периодически;
- загрузка ЭВМ в нормальном режиме поступления сообщений и заданий, а также вероятность перегрузки заданиями различных типов и возможные распределения длительностей перегрузки в реальных условиях;
- влияние пропуска в обработке заданий или сообщений каждого типа и снижения темпа решения определенных задач на функциональную пригодность и другие важные характеристики программного комплекса.

Перечисленные задачи могут быть ***решены экспериментально*** в процессе тестирования завершенного разработкой программного комплекса, однако при этом ***велик риск***, что производительность ЭВМ окажется недостаточной для решения заданной совокупности задач в реальном времени, что отразится на качестве использования системы. Кроме того, не всегда условия испытаний или опытной эксплуатации системы соответствуют режимам массового ее применения. Поэтому при оценивании требуется принимать специальные меры для создания реальных, а также контролируемых, наиболее тяжелых по загрузке условий функционирования комплекса программ и

внешней среды. Такие ***критические ситуации*** могут быть в значительной степени предотвращены в процессе разработки комплекса программ путем расчета длительностей исполнения компонентов по тексту программ, и объединения этих характеристик в соответствии со структурой всего комплекса программ.

Для корректного ***оценивания предельной пропускной способности*** системы с данным программным комплексом необходимо измерять следующие характеристики реализации функциональных программ в процессе разработки:

- экстремальные значения длительностей их исполнения и маршруты, на которых эти значения достигаются;
- среднее значение длительности исполнения каждой функциональной группы программ на всем возможном множестве маршрутов и его дисперсию;
- распределение вероятностей и значений длительности исполнения функциональных групп программ.

Достоверность оценивания пропускной способности системы, с конкретным программным продуктом, зависит от корректности моделирования потоков внешних сообщений, а также от используемых распределений длительности исполнения программ. Для оценивания ресурсной эффективности, при подготовке технического задания и спецификаций требований ***следует согласовывать с заказчиком модель и характеристики внешней среды***, в которой будет применяться комплекс программ, а также динамику приема и передачи данных. Для определения использования комплексами программ временных ресурсов ЭВМ полезно применять рекомендации стандарта **ISO 14756**. Стандарт ориентирован на динамическое оценивание: программных комплексов, операционных систем и вычислительных комплексов, включающих аппаратные и программные средства. Описание метода измерения производительности начинается с имитации пользователей и потоков данных из внешней среды: их случайных характеристик и процессов; функционирования терминалов; установления параметров рабочих нагрузок пользователей и вычислительных средств.

По результатам квалификационного тестирования и испытаний могут быть решены задачи динамического оценивания ресурсной эффективности программного комплекса, что позволяет анализировать факторы, определяющие необходимую пропускную способность

ЭВМ, и разрабатывать меры для приведения ее в соответствие с потребностями. Если предварительно в процессе проектирования производительность системы не оценивалась или определялась слишком грубо, то ***велик риск***, что доработки будут большими или может понадобиться заменить ЭВМ на более быстродействующую. Это обусловлено, как правило, «оптимизмом» разработчиков, что приводит к ***занижению интуитивных оценок длительностей решения функциональных задач*** и возможных предельных интенсивностей потоков внешней информации. Длительная регистрация и накопление значений ресурсной эффективности способствуют выявлению ситуаций, при которых проявляются некоторые дефекты функциональной пригодности.

Тестирование документации на соответствие требованиям к программным комплексам

Эксплуатационная документация должна обеспечивать ***эффективное применение программного комплекса*** и точно отражать его назначение, функции, характеристики и требования, для использования квалифицированными специалистами-пользователями. Для этого эксплуатационные документы необходимо тестировать на полное соответствие выполнения всей ***совокупности требований на программный комплекс***, согласованных между разработчиками и заказчиком. В результате эти документы можно использовать как отдельный, независимый при разработке ***третий эталон и вид тестов,*** реализации требований к функциям и характеристикам программного комплекса. Практическое апробирование документов пользователями при применении комплекса программ, является практическим методом тестирования корректности реализации требований к программному продукту, завершающим полный цикл контроля его качества. Разработчики документов должны обеспечивать комфортное и корректное применение комплекса программ пользователями, на основе ясного и непротиворечивого изложения в документах технологических процедур и операций для его штатного функционирования и получения ***требуемых результатов***.

Организация документирования должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения документов на комплекс программ. Для этого в общем случае должны быть выделены ***специалисты***, которые обязаны планировать, утверждать, выпускать, распространять и сопровождать комплекты апробированных и утвержденных эксплуатационных до-

кументов. Они должны стимулировать разработчиков программных средств, осуществлять непрерывное, регламентированное документирование процессов и результатов своей деятельности, а также контролировать полноту и качество утвержденных эксплуатационных и отчетных документов.

Состав и содержание комплекта документов конкретного программного комплекса, целесообразно адаптировать разработчикам к его особенностям и свойствам ***на основе использования стандартов и типовых структур – шаблонов*** для двух классов продуктов, которые в наибольшей степени различаются особенностями эксплуатации. ***Первый класс*** составляют программные комплексы автоматизированного управления динамическими объектами и процессами в реальном масштабе времени. В процессе их применения допускается минимальное вмешательство пользователями в процедуры управления применением, и необходим, соответственно, небольшой объем эксплуатационных документов, выделяемых из стандартизированного комплекта. Для программных комплексов ***второго класса*** возможно применение пользователями широкого набора процедур управления, которые должны быть регламентированы достаточно полным набором и подробным содержанием документов. Пользователей таких программных комплексов можно разделить на две крупных группы, каждая из которых должна быть обеспечена комплектной эксплуатационной документацией:

- администраторы, подготавливающие программный комплекс к эксплуатации, обеспечивающие их функционирование и использование по прямому назначению;
- операторы – пользователи, реализующие применение программных комплексов в системе, их функционирование, обработку и анализ результатов.

Документация администрирования при эксплуатации системы должна обеспечивать поддержку первичной инсталляции, безопасного функционирования и восстановления программ и данных после сбоев. Администратор системы и программного комплекса должен быть информирован о всех изменениях функционирования устройств системы и внешней среды, могущих привести к сбою или возникновению аварийной ситуации, и предпринимать соответствующие профилактические действия. Для этого требуется полная информация о требованиях программному комплексу, к компонентам системы и внешней среды, которые имеют свои особенности в управлении с по-

мощью специальных программ, поддерживающих администрирование и управление системой. Каждый из документов административного управления ***должен не противоречить международным стандартам*** на коммуникацию, интерфейсы с пользователями и базами данных, на защиту и обеспечение безопасности информации.

Документация для оперативных пользователей программных комплексов в системе, их функционирования, обработки и анализа результатов должна обеспечивать взаимодействие пользователей с различными аппаратно-программными реализациями терминалов. Для этого необходима унификация концепции, архитектуры, функций и методов визуализации пользовательского интерфейса. Типовые ***формы-шаблоны документов*** и процедуры работы с ними, рассматриваемые ***как объекты стандартизации***, относятся в основном к функциональному, оперативному уровню взаимодействия пользователей с системами и программными комплексами.

Стандарты **ISO 15910** и **ISO 18019** являются наиболее современными нормативными документами, регламентирующими процессы ***создания эксплуатационной документации для пользователей сложных программных комплексов***. В них изложены детальные структуры планов документирования, ориентированных на разработчиков документов, соответствующих требованиям на программные комплексы, и процедуры контроля реализации плана. Эксплуатационная документация должна проходить ***тестирование и испытания на достоверность***, которые должны быть спланированы и реализованы на базе требований к функциям и характеристикам, а также к применению эксплуатационных процедур реального программного комплекса. Изложены требования и правила оформления ***твердой копии документов*** и правила структурирования и представления схем компонентов, окружения, иллюстраций и основного текста документов.

Стандарты представляют разработчикам документации метод определения и применения процесса документирования при создании конкретного программного комплекса. Для соответствия стандартам план должен включать спецификацию требований к стилю оформления документов. Стандарты также определяют виды информации, представляемой заказчиком разработчику документации для тестирования, проверки и распространения документации. Стандарты определяют реализацию процесса документирования, описанного в **ISO 12207**, и могут быть ***адаптированы к условиям и требованиям конкретного проекта*** (см. рис. 2.17). Минимальный состав документации определя-

ется заказчиком (например, с использованием **ISO 12207** или **ISO 6592**), что должно быть учтено при разработке плана документирования.

Описание требований эксплуатационной концепции для системы управления и программного комплекса, должно содержать действия пользователя, необходимые для работы с системой, ее связи с существующими системами и процедурами и включает ***описания***:

- конкретной эксплуатационной среды и ее характеристики;
- основных компонентов и функций программного комплекса, системы и связей между ними;
- внешних интерфейсов программного комплекса и системы;
- возможностей, функций и характеристик программного комплекса и системы;
- состава эксплуатационного персонала, его организационной структуры, уровня технической подготовки, обязанностей, взаимодействия;
- форм регистрации обнаруженных дефектов и ошибок;
- соглашений о внесении изменений, возникающих в процессе сопровождения версии программного комплекса;
- концепцию поставки новой или модифицированной версии программного комплекса, эксплуатационный сценарий;
- информацию о взаимодействии пользователей, поставщика, разработчика и предприятия, осуществляющих поддержку программного комплекса, во время эксплуатационного периода.

В обязанности документаторов входит обеспечение плодотворных ***контактов заказчика с разработчиками*** программного комплекса, гарантирующее, понимание сути и требований к выпускаемой продукции и соответствующих ей аудиторий. Документатор должен предпринять соответствующие шаги по сохранению материалов, представленных заказчиком, обеспечить ***защиту информации о требованиях заказчика***. В ряде случаев требуется сохранить конфиденциальность и секретность предоставленных материалов.

План документирования должен включать ***определение аудитории пользователей документации***, уровня образования, квалификации, способностей, подготовки, опыта пользователей и другие характеристики, связанные с содержанием, структурой и использованием документации. ***План*** должен быть официально согласован и утвержден, что подтверждает полный учет и ***тестирование в докумен-***

тах всех требований заказчика к программному комплексу. Должны быть определены процессы тестирования и проверки, выполняемые при разработке документации. План документирования должен быть подготовлен и утвержден до начала разработки документации, чтобы гарантировать согласование всеми сторонами содержания поставленных задач и используемых в проекте методов. После утверждения плана он должен быть доведен до всех заинтересованных сторон, включая разработчиков документации, а также заказчика и субподрядчиков.

Проверка результатов выполнения плана должна проводиться заказчиком с привлечением документаторов. Целью проверки является гарантирование полноты и достоверности, представленных материалов и удовлетворения ими возможности выполнения ***требований заказчика к функциям и характеристикам программного комплекса*** в соответствии с условиями договора и плана документирования. Заказчики должны уделять ***особое внимание структуре, полноте и практичности документации*** в соответствии с планом-проспектом ее содержания. План документирования должен быть проверен и утвержден до начала работ над первым проектом документации.

При ***контроле изменений и сопровождение документации*** должны быть предусмотрены:

- изменения требований и функций программного комплекса, внесенные при разработке и тестировании документации и отраженные в опубликованной документации;
- изменения требований к программному комплексу, внесенные при разработке документации и не отраженные в опубликованной документации, но подлежащие учету в последующей версии;
- изменения программного продукта после публикации – изменения конкретных требований и функций программного комплекса после издания документации;
- изменения документа после публикации – изменения в опубликованной документации, обусловленные изменениями комплекса программ или обнаружением дефектов и ошибок в данной документации.

Документатор должен обеспечить разработку документации так, чтобы допустить возможность внесения в нее корректировок. Для этого необходимо, чтобы разработчики документации получали учтенные копии ***изменений требований***, подтверждающие внесение

соответствующих изменений в данную версию после конкретной даты ее пересмотра.

Тестирование эксплуатационной документации на практичность следует рассматривать как дополнение к проводимым проверкам и анализам достоверности документации. При необходимости в плане документирования следует определить внешнюю среду тестирования, которая должна полностью соответствовать ***эксплуатационной среде конечного пользователя и требованиям к программному комплексу***. При проведении тестирования документации на практичность, необходимо проверить соответствие и актуальность документов конкретному программному комплексу. Для повышения эффективности данного тестирования его необходимо проводить как можно раньше, внося необходимые изменения, как в программный продукт, так и в его документацию. При составлении графика тестирования эксплуатационных документов необходимо учитывать тестирование отдельных компонентов и выполняемых ими функций. При проведении тестирования после завершения разработки следует использовать утвержденную версию программного комплекса. В проведении тестирования документации на практичность должны участвовать представители заказчика.

При тестировании может быть использовано для оценки практичности определения характеристик качества в **ISO 9126** и **ISO 12119**. ***Практичность – применимость***: свойства программного комплекса и документации, отражающие сложность его понимания, изучения и использования для квалифицированных пользователей при применении в указанных условиях. Требования к практичности и ее характеристикам – понятности и простоте использования, зависят от назначения и функций комплекса и могут формализоваться заказчиками набором свойств, необходимых для обеспечения удобной и комфортной эксплуатации. Количественно, простоту использования можно характеризовать требованиями допустимой средней длительности ввода типовых заданий и времени отклика на них. Требования к продолжительности изучения, достаточной для эффективной эксплуатации системы квалифицированными специалистами, могут составлять часы или недели. Для обеспечения полноценного изучения процессов применения программного комплекса специалистами необходима эксплуатационная документация, объем которой существенно зависит от назначения и функций, и может быть задан на основе анализа пре-

цедентов подобных успешных проектов. Для некоторых проектов, подлежащих широкому тиражированию, могут быть желательны адекватные по содержанию электронные учебники, требования к объему и функциям которых целесообразно оценивать по прецедентам. Следует учитывать, что малый объем эксплуатационной документации может снизить качество и полноту использования функций сложного продукта, а очень большой объем – также может ухудшить эксплуатацию из-за трудности выделения из множества второстепенных деталей и освоения, наиболее существенных свойств и особенностей его применения.

В практичности следует учитывать все разнообразие характеристик внешней среды пользователей, на которые может влиять продукт, включая требующуюся подготовку к использованию и оценке результатов функционирования комплекса программ. ***Применимость (практичность) использования программного комплекса и документации*** – понятие достаточно субъективное и трудно формализуемое, однако в итоге зачастую значительно определяющее функциональную пригодность и эффективность применения продукта. В эту группу показателей входят атрибуты с различных сторон отражающие функциональную понятность, удобство освоения или простоту использования. Некоторые характеристики можно оценивать экономическими показателями – затратами труда и времени квалифицированных специалистов на реализацию соответствующих функций.

Понятность: свойства, обеспечивающие пользователю понимание документации, является ли программный комплекс пригодным для его целей, и как его можно использовать для конкретных задач и условий применения. Понятность зависит от качества документации и субъективных впечатлений от функций и характеристик комплекса. Ее можно описать качественно четкостью функциональной концепции, широтой демонстрационных возможностей, полнотой, комплектностью и наглядностью представления в эксплуатационной документации возможных функций и особенностей их реализации. Она должна обеспечиваться корректностью и полнотой описания исходной и результирующей информации, а также всех деталей функций программ для пользователей.

Простота использования: возможность пользователю удобно и комфортно эксплуатировать и управлять программным комплексом.

Аспекты изменяемости, адаптируемости и легкости инсталляции могут быть предпосылками для простоты использования и выбора конкретного продукта. Она соответствует управляемости, устойчивости к ошибкам и согласованности с ожиданиями и навыками пользователей. Эта характеристика учитывает физические и психологические особенности пользователей и отражает уровень комфортности условий эксплуатации, возможность предотвращения ошибок пользователей. Должны обеспечиваться простота управления функциями и достаточный объем параметров управления, реализуемых по умолчанию, информативность сообщений пользователю, наглядность и унифицированность управления экраном, а также доступность изменения функций в соответствии с квалификацией пользователя и минимум операций, необходимых для запуска определенного задания и анализа результатов. Кроме того, удобство использования характеризуется рядом динамических параметров: временем ввода и отклика на задание, длительностью решения типовых задач, временем на регистрацию результатов, которые перекрываются с динамическими характеристиками временной эффективности.

Простоту использования комплексов программ административных систем, в значительной степени, характеризует корректность и адекватность описаний интерактивных директив управления, объем и время ввода заданий, и время ожидания пользователями результатов при их исполнении. Некоторые атрибуты этой характеристики доступны для более полной количественной оценки путем измерения трудоемкости и длительности соответствующих процессов подготовки и обучения квалифицированных пользователей к полноценной и эффективной эксплуатации программного комплекса.

Изучаемость: свойства программного комплекса и документации, обеспечивающие удобное освоение его применения достаточно квалифицированными пользователями. Она может определяться трудоемкостью и длительностью подготовки пользователя к полноценной эксплуатации. Атрибуты изучаемости зависят от возможности предварительного обучения и от совершенствования знаний в процессе эксплуатации, от возможностей оперативной помощи и подсказки при использовании, а так же от полноты, доступности и удобства использования документов, руководств и инструкций по эксплуатации. Качество изучаемости зависит от внутренних свойств и

сложности комплекса программ и документов, а также от субъективных характеристик квалификации конкретных пользователей.

На значения изучаемости существенно влияют демонстрационные возможности справочных средств обучения, качество и объем эксплуатационной документации. Изучаемость можно отражать трудоемкостью и продолжительностью изучения пользователями соответствующей квалификации, методов и инструкций применения программного комплекса для полноценной эксплуатации. Эти атрибуты может характеризовать трудоемкость от единиц до сотен человекочасов и продолжительность от единиц до тысяч часов, необходимых для освоения квалифицированного применения особенно сложных программных комплексов. Естественно, для создания учебных пособий необходимы определенные затраты труда и времени разработчиков, которые в некоторой степени пропорциональны сложности функций продуктов. Эти требования могут влиять на функциональную пригодность и успех внедрения комплекса программ у пользователей, а также значительно различаться в зависимости от его функционального назначения и сферы применения.

Обучение представляет собой процесс обеспечения и сопровождения обучаемого персонала. Приобретение, поставка, разработка, функционирование и сопровождение программных комплексов, в большой степени, зависит от квалификации персонала. Поэтому обязательно должно быть запланировано и осуществлено обучение с целью подготовки работы персонала при приобретении, поставке, разработке или сопровождения программного комплекса. Для достижения высокого качества программной документации она должна рассматриваться как ***неотъемлемая часть процесса создания и тестирования программного комплекса***.

Ряд документов (спецификации, тесты, тексты программ) разрабатывается в процессе создания комплекса программ и их трудно выделить и тестировать как самостоятельные работы. Только около половины работ можно достаточно четко регламентировать (редактирование, печать, нормоконтроль, утверждение и т.д.). Эти обстоятельства привели к большому различию суммарных удельных затрат на страницу эксплуата-ционных документов сложных программных комплексов. Затраты на создание достаточно ***полного комплекта достоверной эксплуатационной документации*** практически пропорциональны размеру комплекса программ. Эти затраты сопутствуют в некоторой степени всем этапам разработки, однако оформление

эксплуатационных документов обычно локализуется в специальном этапе работ. Затраты на разработку комплекта ***эксплуатационной документации*** для сложных программных продуктов, подлежащих длительному сопровождению, обычно определяются затратами на объем текста документов ***ориентировочно*** 5 – 10 страниц на тысячу строк программы.

В ***технологической документации***, обеспечивающей конфигурационное управление и многолетнее сопровождение версий программного комплекса, необходимы требования, тесты, тексты программ и описания алгоритмов. Это приводит к увеличению объема документации на порядок, т.е. может составлять около 100 страниц совокупности документов на тысячу строк программы. Подтверждена наиболее высокая документированность тиражируемых программ реального времени, особенно в тех случаях, когда необходима высокая безопасность и надежность функционирования продукта (до 200 страниц на тысячу строк).

При оценках можно предполагать средний объем ***технологической документации*** ~ 50 – 100 страниц на тысячу строк. При этом затраты на документацию остаются практически пропорциональными размеру комплекса программ. Эта часть документации не подлежит массовому тиражированию и поставке каждому пользователю, что позволяет несколько снижать удельные затраты на ее подготовку. Однако необходимость ее тщательной отработки, и высокого соответствия текущей версии программного продукта, приводит к большим затратам, как при первичном изготовлении технологических документов, так и при их модификации в процессе последующего сопровождения.

Лекция 2.6

ИСПЫТАНИЯ КОМПОНЕНТОВ И КОМПЛЕКСОВ ПРОГРАММ

Организация и процессы испытаний компонентов и комплексов программ

До начала квалификационного тестирования и испытаний компонентов и комплексов программ подлежат проверке и ***паспортизации средства***, обеспечивающие получение эталонных данных, средства формирования тестов от внешних объектов, средства фиксирования и обработки результатов тестирования. При этом важную роль играет оценка и обеспечение методической достоверности результатов испытаний – рис. 2.18. ***Методическая достоверность испытаний программных комплексов определяется следующими факторами:***

- полнотой Программы испытаний и корректностью методик тестирования по охвату возможных сценариев функционирования компонентов и комплекса программ, и исходных требований для программного продукта;
- достоверностью и точностью эталонных значений требований, с которыми должны сравниваться результаты тестирования испытываемого комплекса программ, которые служат опорными при расчете параметров, зафиксированных в техническом задании и спецификациях требований;
- адекватностью и точностью моделей, используемых для формирования сценариев тестов от внешней среды и их реакции на управляющие воздействия программного комплекса;
- точностью и корректностью регистрации и обработки результатов испытаний, а также сравнения полученных данных с требованиями и эталонами технического задания и спецификаций.

Определение задач испытаний предусматривает анализ документов, в которых сформулированы требования к программному комплексу, и выделение необходимых работ, чтобы затем проверять функции и характеристики, отражающиеся этими требованиями.

Испытания компонентов и сложного комплекса программ должны включать:

- организацию и процессы испытаний компонентов и комплекса программ:
  - определение задач, процессов и методической достоверности испытаний компонентов и комплекса программ;
  - подготовку к интеграции компонентов, комплекса программ и аппаратуры системы;
- интеграцию и тестирование комплекса программ вне системы:
  - испытания интерфейсов, компонентов и комплекса программ на соответствие требованиям и эталонам;
  - оценка реализуемости и планирование испытаний комплекса программ в составе системы;
  - анализ полноты и корректности документации на комплекс программ;
- приемо-сдаточные испытания программного комплекса в составе системы:
  - испытания соответствия функций и характеристик качества программного продукта и системы требованиям и эталонам контракта;
  - удостоверение адекватности и качества технологической и эксплуатационной документации программного продукта требованиям на систему;
  - оформление акта о завершении работ и контракта на создание программного продукта и системы;
- опытную эксплуатацию – Альфа и Бета испытания программного продукта в системе и пользователями;
- завершение испытаний на соответствие требованиям и утверждение готовности программного продукта для предъявления заказчику и поставки пользователям;
- внедрение версии программного продукта для применения в системе и обучение пользователей;
- изменения требований к комплексу программ, создание и сопровождение новых версий программного продукта.

Рис. 2.18

По списку требований и эталонов может быть определена матрица, составляющая задачи испытаний, которым присваиваются приоритеты (см. лекцию 2.3). В качестве руководящих принципов для

присвоения приоритетов функциям должны использоваться документы, содержащие формулировки требований. В то же время могут возникать и дополнительные соображения, оказывающие влияние на выбор приоритетов. Новым функциям и характеристикам имеет смысл посвящать повышенное внимание, нежели слегка модифицированным компонентам. Кроме того, необходимо уделять время на проверку проектных решений и структуры комплекса, а также на тестирование новых компонентов. В дополнение к обычному анализу требований необходимо учесть задачи, которые не имеют прямого отношения к документам с требованиями, но ***влияют на процессы тестирования***:

- составление плана и Программы проведения испытаний;
- разработка тестовых сценариев и динамических моделей внешней среды для генерации тестов;
- отладка тестовых сценариев и генераторов тестов;
- верификация, проверка, выявление и исправление дефектов генераторов тестов;
- определение качества тестов и степени реализации функций и характеристик программного продукта, процесса их сборки и использования;
- пересмотр и корректировка пользовательской документации;
- освоение специалистами по тестированию новых технологий и новых инструментальных средств.

После построения такого перечня работ следует его ***ревизовать*** вместе с группой тестирования, и провести через полный жизненный цикл, определяя ***дополнительные компоненты и задачи***, которые должны выполняться для корректных испытаний. Такой перечень является динамическим документом, претерпевающим изменения на протяжении всего жизненного цикла тестирования и испытаний по мере роста понимания того, что должно быть сделано для получения программного продукта высокого качества, ***соответствующего требованиям и эталонам***. Декомпозиция работ часто может выглядеть как иерархический упорядоченный список видов деятельности, в которых каждой задаче присваивается идентификатор. Дескриптор позволяет отслеживать задачу на протяжении всего жизненного цикла разработки и предоставляет возможность связывать измерения трудозатрат или расходов ресурсов с различными задачами. Благодаря этому можно оценивать показатели, характеризующие фактические затраты на протяжении всего проекта.

Факторы, которые следует учитывать перед испытаниями программного комплекса на соответствие требованиям, должны включать:

- идентификатор версии испытываемого программного продукта;
- результаты исправления дефектов, обнаруженных в предыдущей версии, если список дефектов приводится в спецификации, необходимо указать ссылку на требования, функции и характеристики;
- описание внешней среды, используемой при применении программного продукта;
- эксплуатационные документы для пользователей, такие как руководство пользователя, инструкции по инсталляции, особенности версии конкретного продукта.

Функции, характеристики и свойства компонентов и комплекса программ, которые должны тестироваться – это функциональные возможности, отличительные характеристики и риски программного продукта, а также производительность, надежность, безопасность, переносимость, определенные требованиями заказчика. Клюючевым моментом для специалистов по тестированию является то, что если что-то декларировано заказчиком, оно должно быть отражено в требованиях, Программе и плане проведения испытаний с тем, чтобы это протестировать до поставки продукта заказчику. Основным источником, позволяющим фиксировать, что было согласовано с заказчиком, служат ***документы определения требований и эталонов*** (см. лекцию 1.2). Перечень требований должен сопровождаться технико-экономическим обоснованием и контрольной таблицей для вычисления трудозатрат, необходимых для проведения тестирования и испытаний программного комплекса. Каждая позиция этого перечня должна соответствовать конкретным пунктам документов описания требований, в то же время каждой такой позиции должен соответствовать один или большее число тестов, определенных в ***Программе и спецификациях методик тестирования***.

Внутренние испытания компонентов и программного комплекса (***испытания менеджера проекта***), которые зачастую совмещаются с завершением комплексной отладки, должны оформляться документально и могут являться основанием при предъявлении программного продукта заказчику на квалификационные испытания для завершающего оценивания функций и характеристик качества про-

граммного продукта (см. **ISO 12207, ISO 15504, ISO 16326**). Руководитель разработки должен ***оценить уровень реализацию проекта***, состояние комплекса программ, тесты, результаты тестирования и документацию для пользователя, учитывая:

- полноту охвата испытаниями всех требований и эталонов к функциональным компонентам и к комплексу в целом;
- согласованность с требуемыми заказчиком результатами применения программного продукта;
- возможность интеграции и тестирования комплекса программ в составе аппаратуры системы;
- возможность функционирования и сопровождения версии программного продукта в соответствии с требованиями контракта.

Этапы тестирования компонентов и комплекса в целом обычно реализуются с позиции последовательного увеличения функциональной сложности тестов и взаимодействия с объектами внешней среды. Этапы и процессы квалификационного тестирования ***с целью формального удостоверения для заказчика достигнутых характеристик и реализации требований*** к комплексу программ и его компонентам в составе системы регламентированы в стандартах **ISO 12207, ISO 15504**. В них выделены ***основные, функциональные*** этапы реализации квалификационного тестирования и испытаний на соответствие требованиям заказчика (см. рис. 2.18):

- квалификационное тестирование функциональных компонентов и комплекса в целом вне аппаратуры системы;
- интеграция и тестирование программного комплекса в целом в составе аппаратуры системы и полные испытания системы в комплексе с программным продуктом на соответствие исходным требованиям заказчика.

Квалификационное тестирование функциональных компонентов и комплекса в целом вне системы выполняется для того, чтобы предварительно продемонстрировать заказчику, что реализованы все требования контракта и достигнуто необходимое качество функционирования комплекса программ. Квалификационное тестирование должно ***покрывать все требования*** к функциональным компонентам, требования к комплексу и к интерфейсам. Тестирование функциональных компонентов для каждой конфигурации должно показать, что полностью удовлетворены требования к компонентам, которые должны быть реализованы в данной конфигурации. Ответственными за квалификационное тестирование компонентов не должны

быть лица, осуществившие выполнение рабочего проекта или программирование соответствующего компонента. Это не исключает возможность оказания помощи при проведении квалификационного тестирования со стороны лиц, выполнявших рабочий проект или программы, например, путем предоставления тестовых вариантов, основанных на знании ими внутренней реализации функционального компонента или всего комплекса.

Квалификационное тестирование и предварительные испытания должны ***включать работы и объекты***:

- подготовка тестовой версии комплекса программ, содержащей, по крайней мере, два взаимодействующих сложных функциональных компонента, прошедших компонентное тестирование;
- версию комплекса программ, предназначенную для комплексного тестирования, содержащую компоненты, находящиеся под формальным автоматизированным контролем системы управления конфигурацией;
- тестовую версию, собранную из относительно новых функциональных компонентов, которая может быть установлена в тестовой внешней среде таким образом, что она будет стабильно функционировать, получать, поддающиеся интерпретации результаты испытаний;
- заключительную версию, предназначенную для комплексного тестирования и испытаний, содержащую все компоненты, которые войдут в версию, предназначенную для пользователей или установки в системе.

Разработчики должны определить и зарегистрировать процесс подготовки к тестированию, тестовые варианты, сценарии и процедуры, которые должны использоваться для квалификационного тестирования компонента и проследить ***соответствие между тестовыми вариантами и требованиями*** контракта. Они должны подготовить исходные тестовые параметры, необходимые для выполнения тестовых вариантов. Если испытание функционального компонента должно быть засвидетельствовано представителем заказчика, то до его проведения разработчик должен проверить тестовые варианты и тестовые процедуры, чтобы гарантировать, что они полны и точны, и что программный комплекс готов для проведения тестирования в присутствии представителя заказчика.

Результаты этих проверок должны быть ***зарегистрированы в файлах системы управления конфигурацией*** комплекса программ, а тестовые варианты и процедуры соответствующим образом модифицированы для устранения выявленных дефектов. Следует выполнить все необходимые изменения в комплексе программ, предварительно уведомив об этом представителя заказчика, осуществить ***регрессионное тестирование*** в необходимом объеме, модифицировать файлы разработки и другие программные продукты, основываясь на результатах интеграции и тестирования компонентов. Результаты этих действий должны быть включены в отчет для заказчика о результатах проведенных разработчиком предварительных испытаниях комплекса программ. Группа управления проектом должна установить, что выполненное комплексное тестирование достаточно, и дальнейшее продолжение комплексного тестирования, по всей вероятности, не выявит новых дефектов, связанных с интеграцией программных компонентов. Совещание разработчиков и заказчика, принимает решение о том, что критерии завершения квалификационного тестирования вне системы выполнены.

Интеграция и тестирование комплекса программ в составе системы должно содержать их объединение, тестирование полученного в результате комплекса, с целью определения работают ли они корректно вместе, как требуется по контракту. При интеграции ***функциональных компонентов,*** комплексов программ ***реального времени*** целесообразно выделять этапы:

- комплексирование и тестирование функциональных групп программ без взаимодействия с другими компонентами и возможно без подключения к операционной системе реального времени;
- интеграция и тестирование функциональных групп программ с учетом взаимодействия с некоторыми другими компонентами и с базой данных;
- интеграция и тестирование программных компонентов в реальном времени во взаимодействии с другими функциональными компонентами и с компонентами операционной системы и базы данных.

После интегрирования всех откорректированных функциональных компонентов начинаются их тестирование и испытания в составе комплекса программ в целом на соответствие требованиям. Для них наиболее характерны следующие ***этапы интеграции и квалификационного тестирования версии программного продукта в реаль-***

ном времени:

- по данным моделирующего стенда или генераторов динамических тестов, имитирующих отдельные объекты и функционирование внешней среды;
- с имитаторами отдельных объектов внешней среды и с реальными воздействиями от операторов-пользователей;
- в полностью адекватной реальной или имитированной внешней среде и с реальными воздействиями от операторов-пользователей на соответствие требованиям.

Процесс должен продолжаться до тех пор, пока интеграция и тестирование не будут выполнены для всех функциональных компонентов программ и аппаратуры. Тестовые варианты должны покрывать все требования системного уровня, требуемые функции и характеристики программного продукта.

Квалификационное тестирование при испытаниях системы в целом выполняется, чтобы продемонстрировать заказчику, что ***удовлетворены все требования*** технического задания, – функции и характеристики качества соответствуют условиям контракта (см. рис. 2.18). Оно должно покрывать все требования в спецификациях системы и подсистем, а также требования к интерфейсу с внешней средой. Испытания должны включать тестирование на объектной вычислительной системе или на альтернативной модели системы, одобренной представителем заказчика. В процессе ***системного тестирования*** проверяется интеграция отдельных частей, в совокупности составляющих систему в целом. Тестирование на системном уровне обычно проводится специальной группой специалистов заказчика, которая должна провести анализ с целью определения выполнения требований и компонентов функциональности, которые ***вызывают наибольшее количество проблем***. Анализ результатов тестирования может показать, дало ли выполнение тестовых процедур результат в плане выявления ошибок. ***Тест-менеджер несет ответственность*** за проведение тестирования согласно Программе и плану-графику, а также за распределение и перераспределение работ между тестировщиками для разрешения проблем, возникающих в ходе тестирования. Для эффективного выполнения этой функции тест-менеджеру необходимо динамически отслеживать ход тестирования, готовить и анализировать отчеты.

Разработчики должны участвовать в разработке и регистрации процесса подготовки к тестированию, тестовых вариантов, сценариев и тестовых процедур, которые нужно использовать для полного испытания системы, и в прослеживании ***соответствия между тестовыми вариантами и требованиями*** к функциям и характеристикам системы. Каждое проверяемое требование должно соответствовать конкретным, обоснованным характеристикам системы, иметь уникальный для проекта идентификатор, чтобы можно было провести испытание и проследить его выполнение с помощью объективного теста. Для каждого требования должны выбираться квалификационные методы для функциональных подсистем и компонентов программного комплекса, которые необходимо прослеживать в требованиях к системе. Степень детализации требований следует выбирать, учитывая в первую очередь те функции и характеристики качества, которые внесены в условия приемки системы заказчиком, и отдавать ***приоритет*** тем из них, которые заказчик требует обеспечить ***обязательно***.

Все полученные результаты должны быть включены в отчет об испытаниях программного продукта и системы. Если квалификационное тестирование системы должно быть засвидетельствовано представителем заказчика, то до его проведения разработчик должен проверить тестовые варианты и тестовые процедуры, чтобы гарантировать, что они полны и точны и что система готова для проведения тестирования в присутствии представителя заказчика. Испытания должны быть выполнены в соответствии с ***утвержденными заказчиком*** планом, Программой, тестовыми вариантами, сценариями и процедурами.

База данных должна содержать полный набор данных о дефектах и ошибках, обнаруженных во время тестирования программного комплекса. Сводку дефектов, обнаруженных во время испытаний, целесообразно включать в отчетный доклад. В таком случае всем сторонам, заинтересованным в успехе разработки, не потребуется обращаться в базу данных дефектов за сведениями о ходе работ по испытаниям и о качестве программного продукта. Окончательная версия отчета по результатам анализа дефектов, в котором показаны все обнаруженные дефекты и их окончательные корректировки, а также редакция отчета о не устраненных дефектах и ошибках являются необходимыми дополнениями отчетного доклада о результатах испытаний.

Квалификационное тестирование может завершаться ***приемо-сдаточными испытаниями***, которые подразумевают участие заказчика и возможно пользователей. Приемо-сдаточные испытания, как правило, представляют собой подмножество совокупности тестов, использованных на системном уровне. Обнаруженные в ходе приемо-сдаточных испытаний дефекты документируются в отчетах о проблемах, и определяется их приоритеты. Проблемы, устранение которых невозможно осуществить в отведенные на приемо-сдаточные испытания сроки, следует передавать техническому Совету для дальнейшего рассмотрения и оценки. По итогам проведения приемо-сдаточных испытаний должен готовиться отчет, который содержат, краткое описание произведенных работ и оценку полученных результатов тестирования.

Наиболее полным и разносторонним испытаниям должна подвергаться первая версия программного продукта. При ***испытаниях очередных версий программного продукта*** возможны значительные сокращения объемов тестирования апробированных повторно используемых компонентов. Однако комплексные и завершающие испытания каждой новой версии программного продукта, как правило, должны проводиться в достаточном объеме, гарантирующем проверку выполнения ***всех требований модифицированного технического задания***. Для выявления дефектов в процессе эксплуатации серийных образцов в каждом из них должен быть предусмотрен некоторый минимум средств для оперативной проверки функционирования и обнаружения искажений результатов. Эти средства должны позволять фиксировать условия неправильной работы программ и характер проявления дефектов при применении программного продукта. Последующее исправление ошибок должно проводиться специалистами, осуществляющими сопровождение программного продукта.

Установка приоритетов для тестирования конфигураций версий программных продуктов обычно зависит от ***следующих факторов***:

- частота использования: сколько экземпляров данной конфигурации, скорее всего, будет использоваться;
- риск отказа в работе системы: существуют ли особенно ответственные конфигурации программного продукта для важных заказчиков с требованиями минимальных рисков;
- оценка вероятности отказа системы: фиксировались ли в прошлом отказы конкретных конфигураций программного продукта

и как часто.

Любые ***испытания ограничены допустимым количеством и объемом тестирования***, а также длительностью работы комиссии испытателей, поэтому ***не могут гарантировать абсолютную проверку*** программного продукта, на соответствие требованиям к функциям и характеристикам. Для повышения достоверности определения и улучшения оценивания характеристик, после внутренних или квалификационных испытаний, некоторые экземпляры комплексов программ целесообразно передавать пользователям на ***опытную эксплуатацию в типовых условиях***. Это позволяет более глубоко оценить эксплуатационные характеристики созданного комплекса и оперативно устранять некоторые дефекты и ошибки. Опытную эксплуатацию целесообразно проводить разработчиками с участием испытателей-заказчиков и некоторых пользователей, назначаемых заказчиком. Результаты и характеристики качества опытной эксплуатации после внутренних испытаний могут учитываться при проведении заказчиком квалификационных испытаний для их сокращения.

Для заказчика и пользователей доминирующее значение могут иметь номенклатура и особенности реализации некоторых функций комплекса программ, которые, как правило, требуют наибольших затрат на тестирование и определяют основной эффект от применения программного продукта, а также потенциальный уровень спроса на рынке. Если затраты на разработку комплекса программ можно оценивать и прогнозировать с некоторой достоверностью, то эффективность применения и особенно будущий спрос на конкретную версию программного продукта со стороны пользователей априори оценить трудно. Такие оценки могут проводиться на основе специальных маркетинговых исследований и опыта эксплуатации аналогичных комплексов программ или достаточно близких их прототипов.

Представленные выше процессы испытаний компонентов и сложных комплексов программ ориентированы на ***наличие конкретного заказчика*** программного продукта и ограниченное число пользователей, контролируемых заказчиком. Несколько иначе организуются испытания коммерческих пакетов прикладных программ, создаваемых по инициативе предприятия или коллектива разработчиков для продажи широкому кругу пользователей при отсутствии конкретного заказчика. Для таких коммерческих комплексов программ принято проводить ***квалификационные испытания на соответствие требованиям***, формализованным руководителем проекта в два

последовательных этапа – ***Альфа и Бета тестирование***. Они заключаются в нормальной и форсированной (стрессовой) опытной эксплуатации конечными пользователями оформленного программного продукта в соответствии с эксплуатационной документацией и различаются составом, количеством участвующих пользователей и уровнем их квалификации.

При Альфа тестировании привлекаются конечные пользователи, работающие преимущественно в том же предприятии, но не участвовавшие непосредственно в разработке конкретного комплекса программ. Для Бета тестирования привлекаются добровольные пользователи (потенциальные покупатели), которым бесплатно передается версия программного продукта для опытной эксплуатации. При этом особое значение имеет участие компетентных, заинтересованных и доброжелательных пользователей, способных выявить дефекты и своими рекомендациями улучшать качество тестируемых программ. Их деятельность стимулируется бесплатным, ранним получением и освоением нового программного продукта и собственной оценкой его качества. Эти пользователи обязуются сообщать разработчикам сведения о всех выявленных дефектах и ошибках, а также вносить изменения в программы и данные или заменять версии исключительно по указаниям разработчиков. Только после успешной эксплуатации и Бета тестирования ограниченным контингентом пользователей, руководителем проекта или предприятия разработчиков может приниматься решение о передаче программного продукта в продажу для широкого круга пользователей. Обобщенные результаты Бета тестирования могут использоваться как основа для сертификационных испытаний. Количество тестов и длительность обоих этапов тестирования определяются экспертно разработчиками или руководителем проекта в зависимости от сложности комплекса программ и интенсивности потока изменений.

Программа и методики испытаний компонентов и комплексов программ

Оценивание качества и ***соответствия требованиям*** программного продукта при квалификационных и/или приемо-сдаточных испытаниях проводится комиссией заказчика, в которой участвует руководитель (главный менеджер) разработки и некоторые ведущие разработчики, или аттестованной сертификационной лабораторией

(см. **ISO 10006**). ***Комиссия при испытаниях должна руководствоваться следующими документами*** (рис. 2.19):

- утвержденными заказчиком и согласованными с разработчиком контрактом, техническим заданием и спецификациями требований на программный продукт и систему;
- действующими государственными и ведомственными стандартами на жизненный цикл и испытания крупных комплексов программ, на технологическую и эксплуатационную документацию, а также стандартами де-факто, согласованными с заказчиком для использования – профилем стандартов и нормативных документов;
- Программой испытаний по всем требованиям контракта, технического задания и спецификаций;
- методиками испытаний и матрицей тестов, охватывающими каждый раздел требований технического задания, спецификаций и Программы испытаний;
- комплектом адекватной эксплуатационной и технологической документации на программный комплекс.

План испытаний комплекса программ должен описывать порядок квалификационного тестирования функциональных компонентов и подсистем, тестовую внешнюю среду, которая будет использоваться при тестировании, идентифицировать выполняемые тесты и указывать ***план-график*** тестовых действий (см. лекцию 2.3). Для каждой, предполагаемой тестовой реализации или динамического тестового варианта, должны быть указаны:

- используемые версии аппаратных средств;
- интерфейсное оборудование;
- дополнительные внешние устройства;
- генераторы тестовых сценариев или динамических тестовых потоков данных.

Кроме того, в документе должны быть представлены план-график тестирования и ***матрица трассирования тестов к требованиям и эталонам на*** программный продукт и/или на его функциональные компоненты, а также субподрядчики, принимающие участие в тестировании, их роль и ответственность.

Программа испытаний является серией экспериментов и должна разрабатываться с позиции необходимого объема тестирования в процессе проведения испытаний для проверки выполнения всех требований технического задания и соответствия предъявленной документации.

Программа испытаний компонентов и сложных комплексов программ должна включать:

- утвержденные заказчиком требования технического задания, и эталоны на компоненты, комплекс программ и систему;
- комплект стандартов и нормативных документов проекта комплекса программ;
- план испытаний комплекса программ на соответствие требованиям и эталонам;
- Программу испытаний комплекса программ на соответствие утвержденным требованиям и эталонам;
- методики и матрицы трассирования тестов испытаний на соответствие требованиям и Программе испытаний;
- протоколы результатов испытаний программного продукта по разделам Программы испытаний, требований и эталонов;
- комплект адекватной эксплуатационной и технологической документации на программный продукт;
- акт о завершения удачных испытаний и выполнения требований на программный продукт и систему;
- утверждение готовности программного продукта к поставке пользователям;
- завершение эксплуатации версии программного продукта и определение перспективы его развития;
- анализ результатов испытаний, усовершенствование методов и процессов тестирования комплексов программ.

Рис.2.19

Программа испытаний, методики их проведения и оценки результатов, разработанные совместно заказчиком и менеджерами разработки, должны быть согласованы и утверждены. Они должны содержать ***уточнения и детализацию требований*** технического задания и спецификаций для данного программного продукта, а также гарантировать корректную проверку всех заданных функций и характеристик качества. ***Программа испытаний должна содержать следующие четко сформулированные разделы:***

- объект испытаний, его назначение и перечень основных документов, определивших его разработку;
- цель испытаний, с указанием всех требований технического задания, характеристик качества, подлежащих проверке, и ограничений на проведение испытаний программного продукта;

- собственно Программу испытаний, содержащую проверку комплектности спроектированного программного комплекса в соответствие с техническим заданием, план и график проведения тестирования для проверки по всем разделам требований технического задания и дополнительных требований, формализованных отдельными решениями разработчиков и заказчика;
- перечень и содержание методик испытаний, однозначно определяющие все требования, понятия проверяемых функций и характеристик качества, условия и сценарии тестирования, инструментальные средства, используемые для испытаний;
- перечень методик обработки и оценки результатов тестирования программного продукта по каждому разделу Программы испытаний.

Методика испытаний должна содержать: описание организации и матрицу процесса тестирования, тестовые варианты, сценарии и процедуры, которые используются при испытаниях отдельного функционального компонента или комплекса программ в целом. Каждый тест должен иметь уникальный для данного проекта идентификатор; должны быть представлены инструкции для проведения процедур тестирования; описание аппаратуры и инструментальные средства для реализации тестирования, а также инструкции для динамического и регрессионного тестирования. Кроме того, должны быть приведены ссылки на соответствующие проверяемые требования, указаны условия выполнения (конфигурация аппаратуры и компонентов комплекса программ), входные данные, эталонные и ожидаемые результаты, критерии оценки качества результатов, процедура проведения тестирования для каждого тестового сценария, допущения и ограничения.

Большой объем разнородных данных, получаемых при испытаниях сложных комплексов программ, и разнообразие возможных способов их обработки, интерпретации и оценки приводят к тому, что важнейшими факторами становятся ***методики обработки и оценки результатов, а также протоколы проверки по пунктам Программы испытаний.*** В соответствии с методиками испытаний, средства автоматизации должны обеспечивать всю полноту проверок характеристик по каждому разделу методик. ***Результаты испытаний фиксируются в протоколах*** (см. **ISO 12119)** , которые обычно содержат следующие разделы:

- назначение тестирования и раздел требований технического задания, по которому проводились испытания;

- указания разделов методик в соответствии, с которыми проводились испытания, обработка и оценка результатов;
- условия и сценарии проведения тестирования и характеристики исходных данных при динамическом тестировании;
- обобщенные результаты испытаний с оценкой их на соответствие требованиям технического задания и другим руководящим документам, а также технической и эксплуатационной документации;
- описание отличий тестовой и реальной эксплуатационной среды;
- описание обнаруженных дефектов и ошибок и рекомендуемых улучшений в испытываемом комплексе программ;
- выводы о результатах испытаний и о соответствии созданного программного комплекса или его функционального компонента определенному разделу требований технического задания и исходных спецификаций.

Протоколы по всей программе испытаний ***обобщаются в акте,*** в результате чего делается ***заключение о соответствии системы требованиям*** заказчика и о завершении работы с положительным или отрицательным итогом. При выполнении всех требований технического задания заказчик обязан принять программный продукт, и проект считается завершенным.

Завершение испытаний и внедрение версий программных продуктов

Завершение испытаний программного комплекса в составе системы должно зафиксировать следующие ***процессы и результаты***:

- завершена разработка всех функций комплекса программ и исправление всех выявленных ошибок;
- все функциональные компоненты размещены под формальный автоматизированный контроль системы управления конфигурацией компонентов проекта;
- завершено компонентное тестирование всех функций и исправление всех выявленных дефектов, версии программного продукта;
- подготовлена полная версия программного комплекса с контролируемыми изменениями после испытаний;
- версия программного комплекса, переданная на испытания, сопровождается технологической и эксплуатационной документаци-

ей, перечнем изменений, содержит список отчетов о дефектах, которые исправлены в версии;

- предоставлена контролируемая полная версия программного комплекса, которая установлена в тестовой среде, стабильно функционирует и позволяет получать поддающиеся интерпретации результаты испытаний;
- проведены совещания менеджеров и специалистов, посвященные управлению незавершенными работами по устранению дефектов и оценки времени для исправления дефектов;
- в предшествующие несколько недель не произошло ни одного сбоя, остановки, неожиданного прекращения процесса или другой аномалии функционирования комплекса программ на объектной ЭВМ в системе;
- анализ функционирования показал, что комплекс программ достиг приемлемого уровня качества, стабильности, надежности и безопасности;
- оценки покрытия требований допустимых рисков показали, что все риски сокращены до допустимых пределов;
- менеджеры и группа управления проектом системы установила, что программный продукт, как это определено в ходе заключительного цикла испытаний, удовлетворяет требованиям заказчика и пользователей;
- проведено совещание менеджеров разработки с заказчиком и установлено, что ***критерии завершения испытаний программного продукта выполнены***.

Завершаются испытания предъявлением заказчику на утверждение ***комплекта документов***, содержащих ***результаты комплексных испытаний*** версии программного продукта:

- откорректированные тексты программ и данных на языке программирования и в объектном коде, а также полные спецификации требований на программные компоненты и комплекс в целом после полного завершения тестирования и испытаний;
- Программу испытаний программного продукта по всем требованиям технического задания;
- комплект методик испытаний и обработки результатов по всем разделам Программы испытаний;
- тесты, сценарии и генераторы тестовых данных, использованные для испытаний программных компонентов и версии продукта в целом;

- результаты и протоколы квалификационного тестирования, функциональные и конструктивные характеристики программного продукта в реальной внешней среде;
- отчет о подтверждении заданного качества, полные характеристики достигнутого качества функционирования, а также степени покрытия тестами спецификации требований к программному продукту;
- план, методики и средства автоматизации обучения заказчика и пользователей применению испытанной версии программного продукта;
- комплект эксплуатационной документации, описание программного продукта и руководство пользователя в соответствии с условиями контракта;
- технические условия на версию программного продукта, базу данных управления конфигурацией и эксплуатационную документацию для тиражирования и серийного производства;
- руководство по генерации и инсталляции пользовательских версий и загрузке базы данных в соответствии с условиями и характеристиками внешней среды;
- отчет о технико-экономических показателях завершенного проекта версии программного продукта, выполнении планов и использованных ресурсах;
- ***акт о завершении испытаний*** и готовности к поставке и/или предъявлению для сертификационных испытаний версии программного продукта.

Чтобы исключить напрасную трату времени на более поздних этапах, целесообразно подготовить шаблоны этих документов на этапе составления плана проведения испытаний. Полезно заранее подготовить шаблон отчета по результатам тестирования и заполнять его тестовыми данными по мере их готовности. Кроме того, шаблоны проекта тестов могут ускорить работы по их проектированию. Определение и согласование с заказчиком на раннем этапе жизненного цикла состава документов при завершении испытаний, может упростить построение плана работ.

На завершающих этапах испытаний, когда, как правило, на группу тестирования оказывают давление руководители для ускорения с окончанием работ, разработчикам полезно иметь в своем распоряжении ***документ***, в котором оговариваются все требования и до-

полнения заказчика, которые планировалось испытать. Специалисты склонны забывать, какие соглашения и требования были достигнуты, а также планы проведения испытаний. Если проблема возникнет в области, которая не подвергалась испытаниям в соответствии планом и Программой, она должна быть устранена и проверена после выпуска очередной версии программного продукта. Поскольку стоимость исправления дефектов после сдачи программного продукта в эксплуатации велика, важно получать по возможности точную оценку рисков, прежде чем принимать решение проводить дополнительных испытаний того или иного свойства или конфигурации версии программного продукта.

Отчетный доклад о результатах испытаний должен содержать перечень ***всех не устраненных дефектов***, с соглашением и планом того, будут ли они исправлены в более поздних версиях или же их исправление откладывается на неопределенное время. Дальнейшие версии программного продукта могут содержать не устраненные дефекты до тех пор, пока они не будут исправлены. Необходимо вести наблюдение за такими дефектами, чтобы не тратить дополнительные усилия на их выявление в будущих версиях продукта. В плане проведения испытаний будущих версий должны присутствовать ссылки на список не устраненных дефектов. Это позволит тестировщикам учитывать, что их ожидает во время испытаний новой версии программного продукта.

Критерий выхода из испытаний и утверждения готовности версии программного продукта для поставки, которые могут использоваться для принятия решения о прекращении испытаний:

- завершены все запланированные циклы тестирования, и план проведения испытаний выполнен, тестовое покрытие лучше, чем в случае, когда просто не хватило времени для доведения тестирования до логического конца;
- профиль устраненных дефектов соответствует критерию выхода из испытаний и возможное количество не устраненных ошибок на тысячу строк программного кода достигнуто;
- время, отведенное на тестирование и испытания, истекло, хотя неизвестно достигнутое качество программного продукта.

Принимая решение прекратить работы по испытаниям, следует учитывать несколько факторов, которые играют важную роль при ***оценке неготовности программного продукта*** к поставкам:

- количество дефектов, обнаруженных в процессе тестирова-

ния, которые остаются неисправленными;

- общее количество возможных предсказуемых дефектов, которые не были исправлены;
- процентное отношение количества тестов, которые завершились успешным исходом и устранением дефектов, к числу всех запланированных тестов;
- количество тестов, реализация которых не может быть выполнена, поскольку они заблокированы дефектами.

С целью выработки ***оценки готовности версии*** отлаженного программного продукта следует проводить специальные ***совещания менеджеров - разработчиков и заказчика***. В некоторых предприятиях оценку готовности определяет группа тестирования; в других организациях совещание проводит руководитель проекта системы, либо руководитель разработки и заказчик программного продукта. В любом случае в задачу группы тестирования и испытаний входит представление результатов и выработка ***рекомендаций относительно готовности продукта к поставкам***. Хотя оценка готовности может проводиться формально, в любом случае следует фиксировать имена участников и принятое ими решение, касающееся поставок программного продукта. Чтобы оценка готовности содержала результаты и рекомендации предприятия, проводившего испытания, эта оценка должна содержать ссылку на отчетный доклад по результатам испытаний.

Критерии успешного или неудачного прохождения испытаний и отдельных тестов для программных компонентов должны быть включены в тестовые сценарии, в связи, с чем их следует помещать в план проведения испытаний. С самого начала проекта необходимо определить, что следует считать успешным прохождением стадии испытаний, и ***соответствия исходным требованиям и эталонам*** заказчика, когда можно прекратить испытания продукта и начать его поставку. В некоторых случаях критерий выхода из испытаний определяется в Программе испытаний или в документе, который содержит формулировки требований, предъявляемых к программному продукту. Вне зависимости от того, кто служит источником этих критериев, следует давать их четкую формулировку в плане и Программе проведения испытаний.

После завершения испытаний новой версий программного продукта, обычно осуществляется процесс ее внедрение для при-

менения (см. рис. 2.18). Это производится, как правило, в два этапа; силами разработчиков версии в целях обкатки, проверки и выявления ошибок в изменениях на этапе опытной эксплуатации, и посредством использования специализированных коллективов сопровождения для тиражирования и распространения версий программного продукта. Основные обязанности специалистов сводятся к передаче физических носителей с текстами программ и комплектом эксплуатационной документации, а также к проведению консультаций для выделенной группы специалистов заказчика и пользователей. Разработчики и испытатели в этом случае получают возможность непосредственно ***контролировать работу пользователей*** с системой и документацией, что обеспечивает высокую оперативность отработки замечаний и рекламаций, формирование квалифицированных предложений для изменений, оценку эффективности применения версии программного продукта. Кроме того, должны разрабатываться учебно-методический план, подготавливаться учебные пособия, необходимые для обучения пользователей, а также проводиться обучение выделенной группы специалистов, ответственных за последующее обучение коллективов пользователей.

В процессе жизненного цикла большое значение имеет ***история эксплуатации программного продукта***, развития его версий и соответствующая документация. Еще на стадии проектирования первой версии могут возникать ***идеи совершенствования комплекса программ***, которые в то время невозможно реализовать из-за высокой стоимости, ограниченных сроков проектирования или по иным причинам. Идеи изменения могут быть направлены на коренное улучшение функциональных возможностей программ или некоторые «косметические» улучшения реализуемых функций. Идеи небольших корректировок программ целесообразно накапливать отдельно от предложений по существенному совершенствованию программного продукта. Таким образом, должен создаваться ***документ – исходные данные для изменения требований*** и планирования доработок в процессе сопровождения, содержащий разделы:

- выявленные дефекты и ошибки в программном продукте;
- предложения по совершенствованию функций и улучшению качества эксплуатируемых версий программного продукта;
- идеи и предполагаемая экономическая эффективность коренной модернизации, расширения функций и/или улучшения характеристик версии программного продукта.

После внесения изменений в требования специалисты разрабатывают и тестируют конкретные корректировки пилотного варианта программного продукта. Исходными данными для проведения работы при внесении изменений должны быть: версия программного продукта; согласованные с заказчиком предложения о модификации; согласованные документы на реализацию корректировок; отчет о влиянии корректировок и выходные результаты работы по анализу изменений. Сопроводитель должен провести проверки каждого внесенного изменения совместно с заказчиком, утвердившим изменение в целях подтверждения ***целостности и работоспособности измененного программного продукта***. Он должен получить согласование и подтверждение того, что внесенные ***изменения удовлетворяют требованиям заказчика***, установленным в дополнительном договоре, посредством вспомогательного процесса контроля качества, проведения аудита функциональной и физической конфигурации. Результатами данной работы являются: ***новая версия*** программного продукта, включающая в себя принятые изменения требований; отклоненные изменения; отчет о приемке версии; отчеты о проверках и аудитах; отчет о квалификационном тестировании откорректированного программного продукта.

Снятие программного продукта с эксплуатации и развития версий должно быть подготовлено анализом, обосновывающим это решение. В анализе следует определить и экономически обосновать: возможность сохранения устаревшей версии комплекса программ, а также необходимость создания и применения новой версии программного продукта. При снятии программного продукта с сопровождения следует определить необходимые для этого действия, а затем разработать и документально оформить этапы работ, обеспечивающие их эффективное выполнение. Должны быть предусмотрены возможности ***доступа к полным архивным документам*** снятого с сопровождения программного продукта.

Специалисты, выполняющие снятие версий программного продукта с сопровождения и эксплуатации, должны разработать план, предупредить пользователей об этом, провести соответствующее обучение персонала, уведомить всех заинтересованных субъектов о завершении сопровождения и архивировать соответствующие данные. Должен быть разработан, документально оформлен и реализован план снятия с эксплуатации и прекращения активной поддержки про-

граммного продукта сопровождающим предприятием. К запланированным работам целесообразно привлечь пользователей.

Перед прекращением сопровождения следует определить влияние снятия программного продукта с сопровождения на пользователей, установить программный продукт, заменяющий снимаемый (при его наличии) и определить обязанности по любым вопросам последующей поддержки применения версий программного продукта. Пользователи должны получить уведомление о планах и работах по снятию с сопровождения и эксплуатации. В содержание уведомления необходимо включить: описание заменяющей или модернизированной версии программного продукта с указанием даты ее доступности для пользователей; объяснение того, почему прежний программный продукт нельзя больше поддерживать и применять.

Для плавного перехода к новой версии программного продукта в некоторых случаях должна быть обеспечена параллельная эксплуатация прежнего и нового программных продуктов. Следует провести необходимое ***обучение пользователей*** новой версии в соответствии с условиями договора. После выполнения запланированного снятия с эксплуатации должно быть послано соответствующее уведомление всем заинтересованным сторонам. Все, связанное с прежней версией: документы требований, разработки, тестирования и испытаний, журналы регистрации и программы должно быть помещено в архивы под конфигурационное управление (см. лекцию 2.7). Данные, использованные или связанные со снятым с эксплуатации программным продуктом, следует сохранять доступными, для аудиторской проверки. Целесообразно также сохранять старые версии комплекса программ и некоторые данные, полученные при решении предыдущих задач в качестве тестов.

По окончании разработки версии программного продукта специалистам полезно ***изучить ход выполнения Программы испытаний***, чтобы определить, каким образом можно ее усовершенствовать, и в следующем проекте получить преимущества. Тестировщики должны постоянно изучать данные, получаемые специалистами контроля качества, и следовать их рекомендациям по исправлению дефектов. Кроме того, на протяжении всего жизненного цикла комплекса программ целесообразно ***документировать полученные уроки*** и оценивать их на каждом промежуточном рубеже тестирования. Полученные уроки, оценки измерений характеристик программного продукта, любые связанные с этим действия по устранению недостатков

или по совершенствованию работ, должны документироваться на протяжении всего процесса тестирования в централизованной базе данных ***конфигурационного управления требованиями и тестами***. Наличие отвечающей современным требованиям базы данных, содержащей спецификации требований, применявшиеся тесты, важные проблемы и полученные уроки, позволяет специалистам, занятым в проекте, следить за ходом работ и состоянием проблем вплоть до его завершения.

Тест-менеджер должен способствовать тому, чтобы информация о проблемах и полученных уроках рассматривалась, как возможность усовершенствования процессов. В каждой записи негативного события необходимо указать возможный способ усовершенствования процесса тестирования или корректировки его результатов. Предложениям по устранению недостатков должны предшествовать серьезный анализ и тщательная проверка. Для каждого урока желательно указывать измерение, ***демонстрирующее потенциальный выигрыш*** (сэкономленные часы труда), если бы было применено определенное действие по совершенствованию процесса. После завершения тестирования специалисты должны сопоставить фактическую реализацию Программы тестирования с запланированными критериями.

Усовершенствование процессов тестирования целесообразно выполнять итеративно. Группе тестирования следует выяснить, повторялись ли те или иные дефекты и ошибки, и подтвердить, были ли упущены из виду какие-либо предложения по совершенствованию испытаний. При изучении Программы испытаний следует концентрировать внимание на оценке того, удовлетворяет ли комплекс программ критериям завершения тестирования, и готов ли он к поставкам. Группа тестирования должна освоить непрерывный итерационный процесс ***изучения полученных уроков как часть развития культуры тестирования***. Такой процесс должен поощрять тестировщиков предлагать меры по устранению недостатков, когда эти действия способны существенно повлиять на выполнение Программы испытаний.

Финансирование тестирования и испытаний программного продукта целесообразно определять специальным разделом договора между разработчиком и заказчиком на разработку версии комплекса программ. В техническом задании и в контракте следует четко определять порядок квалификации видов и причин изменений в программах при испытаниях, а также распределение ответственности за

их инициализацию, реализацию и финансирование. Выявленные ошибки в компонентах и комплексе программ, которые искажают реализацию функций, согласованных с заказчиком в контракте и требованиях спецификаций, а также отраженные в документации на версию, должны устраняться за счет разработчика. Модификацию и расширение функций и характеристик компонентов или создание новых версий программного продукта, ранее не отраженных в требованиях технического задания и контракте с заказчиком, следует квалифицировать как дополнительную работу с соответствующим финансированием заказчиком.

После передачи версии программного продукта в эксплуатацию затраты ресурсов на обнаружение и первичную квалификацию дефектов несут в основном непосредственные пользователи. На разработчиков (поставщиков) программного продукта возлагаются затраты на анализ и локализацию источников и причин дефектов и их устранение. Эти затраты зависят от характеристик выявляемых дефектов, от масштаба комплекса программ, организации и технологии его разработки, инструментальной оснащенности сопровождения, квалификации специалистов, а также от тиража и активности применения данного программного продукта. Априори перечисленные факторы прогнозировать невозможно и оценки этой составляющей затрат целесообразно проводить по результатам начальных этапов сопровождения и модификаций первых версий. Затраты на тиражирование и адаптацию к параметрам среды пользователей зависят от широты распространения программного продукта и могут оцениваться по типовым прецедентам аналогичных проектов или предшествующих версий. Гибкость и кажущаяся простота изменения программ значительно затрудняют оценки необходимого, иногда весьма значительного, финансирования на тестирование и испытания, а также на сопровождение и проведение жесткого конфигурационного управления версиями программного продукта.

При сопоставлении результатов оценивания функций и характеристик качества с требованиями технического задания и спецификаций, разработчик или поставщик ***обязан удовлетворять требования заказчика только в пределах согласованных параметров*** модели внешней среды и системы. Оценивание функционирования комплекса программ за этими пределам должно дополнительно согласовываться испытателями с разработчиком. При этом не выполнение требований может квалифицироваться как их расширение за пределы, ограниченные контрактом, и не учитываться при оценивании заказ-

чиком характеристик качества программного продукта, или как дополнительные работы, подлежащие соответствующему финансированию со стороны заказчика, для доработки комплекса программ с целью удовлетворения этих требований.

Лекция 2.7

УПРАВЛЕНИЕ КОНФИГУРАЦИЕЙ И СЕРТИФИКАЦИЯ КОМПОНЕНТОВ И КОМПЛЕКСОВ ПРОГРАММ

Задачи управления конфигурацией требований и тестов компонентов и комплексов программ

Основной задачей управления конфигурацией требований и тестов является документальное оформление и обеспечение полной наглядности текущей конфигурации состояния производства и выполнения требований к функциональным характеристикам компонентов и комплекса программ. При этом требования и тесты должны рассматриваться совместно как две формы описания функций и характеристик компонентов и программного комплекса. Другая задача заключается в том, чтобы все лица, работающие над проектом, в любой момент его жизненного цикла использовали достоверную и точную информацию о состоянии требований, компонентов и их реализацией. Управление конфигурацией следует организовать так, чтобы каждый специалист знал свои обязанности и имел достаточно независимости и полномочий для выполнения поставленных задач. Эта управленческая дисциплина, использует техническое и административное руководство при разработке, производстве и поддержке компонентов конфигурации проектов – рис. 2.20.

Четкая, иерархическая структура документов комплексов программ позволяет использовать конфигурационное управление для решения задач ***синтеза – конструирования и модификации конфигураций требований к программному комплексу***, состава и взаимодействия компонентов и тестов в процессе ***сборки и формирования его версий***. Регистрация и учет истории этого процесса обеспечивает возможность его контроля и пошагового восстановления выполненных изменений (отката) для выявления вторичных дефектов, внесенных в процессе тестирования и разработки корректировок компонентов и версии комплекса программ.

Управление конфигурацией и сертификация компонентов и комплексов программ должны включать:

- задачи управления конфигурацией требований и тестов компонентов и комплексов программ:
  - организацию специалистов и процессов управления конигурацией компонентов и комплексов программ;
  - конфигурационную идентификацию и учет версий комнентов и комплексов программ;
  - управление запросами на изменения версий компонентов и комплексов программ;
  - реализацию корректировок версий компонентов и комплексов программ;
- методы, процессы и средства управления конфигурацией требований и тестов компонентов и комплексов программ:
  - контроль корректности изменений конфигурации версий компонентов и комплексов программ;
  - сборку и формирование версий конфигурации программного продукта;
  - методику оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке требований;
  - создание отчетов о состоянии и изменениях конфигурации версий программного продукта;
  - утверждение и выпуск версий конфигурации программного продукта;
  - архивирование и тиражирование версий программного продукта и документов
- управление качеством и сертификацией процессов производства программных продуктов;
- управление качеством и сертификацией программных продуктов;
- сертификацию руководителей и специалистов, управляющих конфигурацией компонентов и комплексов программ.

Рис. 2.20

Такие дефекты обычно обусловлены одновременным, не скоординированным внесением групп изменений или потерей некоторых изменений требований в определенной версии. Это возможно при одновременной разработке или корректировке требований или тестов различных версий программных компонентов, предназначенных для

использования в различных, но определенных версиях сложных комплексов программ.

В ***стандартах, регламентирующих конфигурационное управление,*** представлена широкая группа процедур, составляющих эти процессы (см. Приложение 1). Предприятие или подразделение, в зависимости от стоящей перед ними задачи, могут выбрать соответствующую подгруппу процессов и процедур для выполнения своей конкретной цели. Их множество сконструировано в стандартах так, что возможна их ***адаптация в соответствии с характеристиками проектов и внешней среды системы***. Процесс адаптации является обычно процессом исключения из рекомендаций стандарта процедур, видов деятельности и задач, не применимых в конкретном проекте. Выполнение процесса или процедуры является законченным, когда решены все требуемые задачи в соответствии с установленными заранее критериями и требованиями, специфицированными в конкретной методике или контракте для успешного использования при разработке изменений проекта. В стандарте **ISO 15846** излагаются ***подробные рекомендации*** по реализации базовых требований стандарта **ISO 12207** по управлению конфигурацией, которые, в частности, могут быть адаптированы и применены для управления изменениями требований программного комплекса, включающие:

- подготовку процесса управления требованиями к программному комплексу, компонентам и тестам;
- определение конфигурации требований, тестов и их изменений;
- контроль изменения конфигурации требований и тестов к компонентам и комплексу программ;
- учет состояния конфигурации требований и тестов комплекса программ.

Для описания политики предприятия, видов его деятельности и правил, связанных с процессом управления конфигурацией, следует использовать документированные в стандартах процедуры. Эти виды деятельности в области управления конфигурацией и степень их применения, характерные для конкретного проекта комплекса программ должны быть определены в ***Руководстве по управлению конфигурацией требований и тестов***. Для того чтобы управление конфигурацией требований было эффективным, следует определить его организационную структуру. Эта структура обычно ориентируется на проект

и при необходимости адаптируется, чтобы отвечать потребностям различных этапов жизненного цикла.

Процесс управления конфигурацией требований и тестов включает административные и технические процедуры на всем протяжении жизненного цикла программных комплексов для:

- обозначения, определения и установления состояния компонентов и версии программного комплекса;
- управления изменениями и выпуском компонентов и версий программных продуктов;
- описания и сообщения о состояниях компонентов и комплекса программ и заявок на внесение изменений в них;
- обеспечения полноты, совместимости и правильности описания состояний комплекса программ;
- управления хранением, обращением и поставкой программных продуктов.

Четкая организация и автоматизация этого процесса позволяют избегать множества вторичных ошибок, обусловленных недостаточной координацией проводимых корректировок и формирования новых версий сложных программных комплексов и компонентов коллективом специалистов. Этому должна способствовать утвержденная ***иерархия принятия решений на изменения*** компонентов и комплекса в целом должностными лицами проекта (см. таблицу 1), поддержанная методами и средствами защиты от несанкционированного доступа при выполнении корректировок специалистами различной квалификации и права доступа на разных уровнях проекта. Каждый вариант компонента или комплекса программ и изменений требований к ним должен соответствовать правилам идентификации (см. рис. 2.20). В результате с использованием ограниченной и упорядоченной системы символов должна создаваться ***база для однозначного выбора и манипулирования*** вариантами компонентов или версиями комплексов программ и тестов для процессов прослеживания их изменений.

Конфигурационное управление включает действия и средства, позволяющие устанавливать категории, статус и личности руководителей, которые ***правомочны, ответственно определять целесообразность и эффективность изменений***, а также техническую реализуемость корректируемых версий с учетом ограничений бюджетов и сроков. При анализе и селекции изменений требований важен точный

учет степени влияния каждого изменения на все остальные компоненты и на их основные характеристики качества. Поэтому решения об изменениях должны приниматься на достаточно высоком уровне ответственного руководства проектом, способным оценить их влияние на концептуальную целостность и качество всей системы (см. таблицу 1).

Изменения небольших компонентов и модулей может подвергаться наименее формализованному и защищенному от случайностей конфигурационному управлению. Их изменения в процессе тестирования обычно не требуют санкции руководителей проекта. Версии функциональных групп программ и комплекса программ в целом могут корректироваться только с разрешения руководителей соответствующего ранга. Тем самым должны предотвращаться несогласованные и несанкционированные изменения, способные снизить качество и целостность версий программного продукта. Изменения этих версий допускаются пользователями или поставщиками в пределах, ограниченных эксплуатационными документами по адаптации к характеристикам и особенностям внешней среды и условий применения версии конкретного программного продукта.

Концепция конфигурационного управления конкретным проектом должна предусматривать возможность ***анализа изменений иерархической структуры – конфигурации***, программного комплекса и его компонентов как сверху вниз, так и снизу вверх (см. лекцию 2.3). Первая задача состоит в обеспечении пошаговой детализации сверху вниз возможных причин конкретных дефектов (проявлений ***вторичных ошибок***) или неэффективности функционирования программы, для обнаружения их источника (***первичной ошибки***). Вторая задача при движении снизу вверх должна обеспечивать проверку корректности взаимодействия связанных корректировок и сохранения концептуальной целостности и качества комплекса программ и/или его компонентов.

Базовые версии компонентов и комплексов программ, а также тестов должны регистрироваться в контролируемых библиотеках и позволять ссылаться, управлять и прослеживать изменения их версий и конфигурации. Они должны быть защищены от внесения любых несанкционированных изменений. После проведения работ по реализации и контролю совокупности изменений должна быть разработана и зафиксирована очередная версия, производная от ранее ус-

тановленной базовой версии. Работы ***по прослеживанию изменений*** должны включать:

- подтверждение того, что идентифицированы затронутые изменениями компоненты конфигурации;
- оценку воздействия изменений на требования качества, надежности и безопасности и на обеспечение обратной связи к процессу оценки качества функционирования системы;
- оценку дефектов или предлагаемых изменений требований и тестов, и решения о действиях, которые следует предпринять для их коррекции;
- обеспечение обратной связи от отчетов по дефектам и тестам, контроля изменений и корректирующих действий к затронутым изменениями процессам и объектам.

После оценки предложения об изменении требования и/или теста уполномоченное лицо или группа лиц Совета по управлению конфигурацией должны проанализировать документированные оценки и решить вопрос об ***утверждении или не утверждении изменений***. Предпосылкой правильного ведения отчетности о статусе конфигурации является корректная идентификация и управление изменениями (см. рис. 2.20). В системе отчетности о статусе конфигурации должна представляться информация для руководства процессом управления конфигурацией и связанной с ним деятельностью.

Методы, процессы и средства управления конфигурацией требований и тестов компонентов и комплексов программ

Технической основой управления конфигурацией являются ***системы управления базами данных*** (СУБД), адекватные целям и функциям проектов, ***структурированные*** по целям, назначению и содержанию данных в выделенных подсистемах (рис. 2.21). Они должны обеспечивать возможность управления организационной и проектной деятельностью коллектива специалистов, универсальное хранилище в них необходимых данных, с инструментами наполнения, корректировки, поиска и контроля информации, соответствующей их профессиональной деятельности.

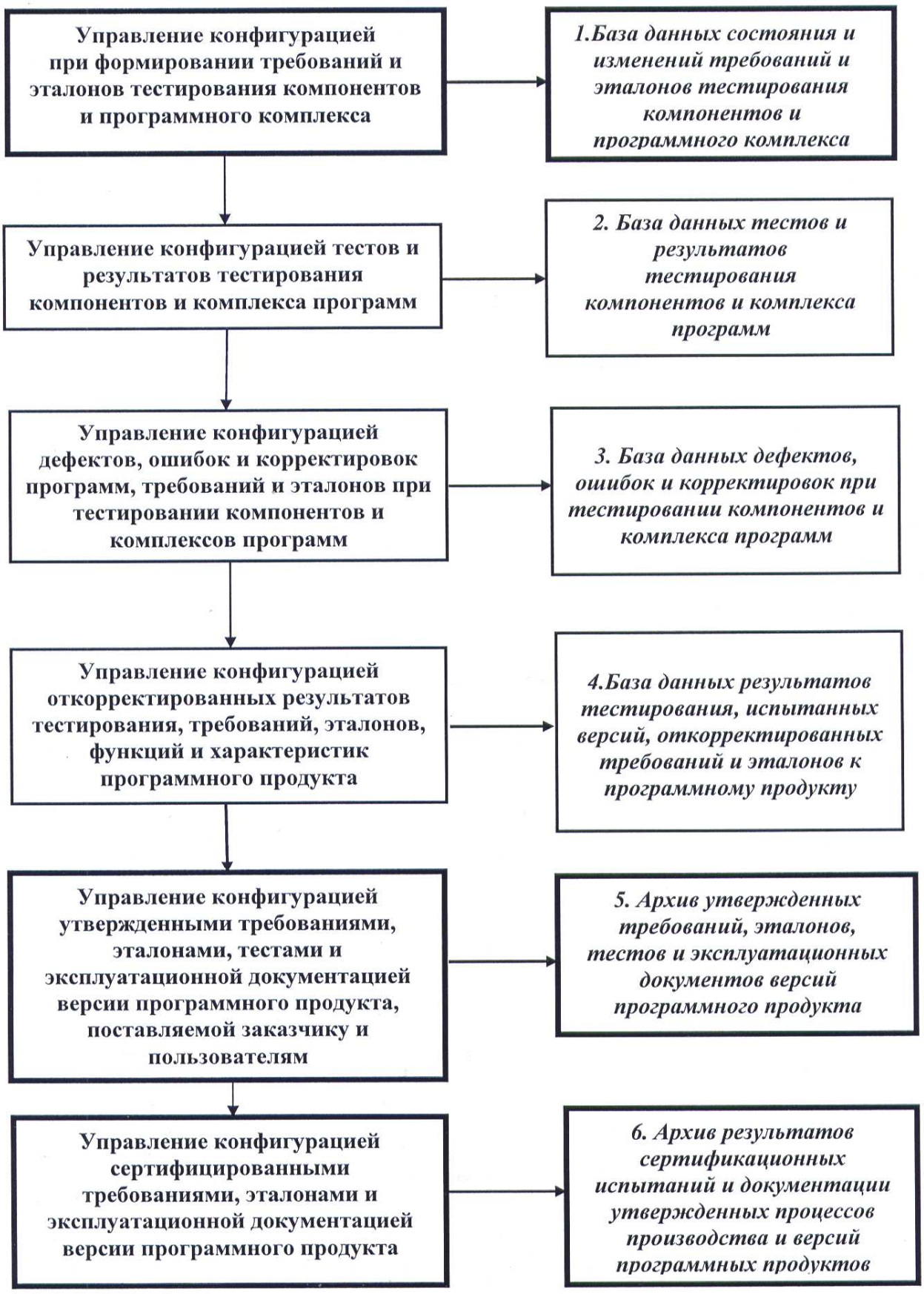

Управление конфигурацией при формировании требований и эталонов тестирования компонентов и программного комплекса
1.База данных состояния и изменений требований и эталонов тестирования компонентов и программного комплекса
Управление конфигурацией тестов и результатов тестирования компонентов и комплекса программ
2. База данных тестов и результатов тестирования компонентов и комплекса программ
Управление конфигурацией дефектов, ошибок и корректировок программ, требований и эталонов при тестировании компонентов и комплексов программ
3. База данных дефектов, ошибок и корректировок при тестировании компонентов и комплекса программ
Управление конфигурацией откорректированных результатов тестирования, требований, эталонов, функций и характеристик программного продукта
4.База данных результатов тестирования, испытанных версий, откорректированных требований и эталонов к программному продукту
Управление конфигурацией утвержденными требованиями, эталонами, тестами и эксплуатационной документацией версии программного продукта, поставляемой заказчику и пользователям
5. Архив утвержденных требований, эталонов, тестов и эксплуатационных документов версий программного продукта
Управление конфигурацией сертифицированными требованиями, эталонами и эксплуатационной документацией версии программного продукта
6. Архив результатов сертификационных испытаний и документации утвержденных процессов производства и версий программных продуктов

Рис. 2.21

Должны быть упорядочены ***деловые коммуникации*** между специалистами разных категорий, управление динамическими процессами выполнения изменений и транспортировки корректировок между подсистемами в соответствии с целями их использования специалистами.

Должна быть разработана ***архитектура системы технологического обеспечения управления конфигурацией, а также Руководство по ее применению***, адаптирована выбранная СУБД для управления основными взаимодействующими подсистемами базы данных, с учетом класса и масштаба предполагаемого комплекса программ и количества компонентов. По мере развития жизненного цикла комплекса программ, подсистемы базы данных управления конфигурацией должны поэтапно заполняться реальными данными от заказчика и разработчиков соответствующих квалификаций, и контролироваться менеджерами проекта. При этом следует управлять динамикой процессов реализации процедур модификации, регистрировать реальное использование ресурсов специалистов, текущее время и график выполнения процедур развития проекта и оформления изменений в подсистемах базы данных.

Процессы и структура базы данных управления конфигурацией должны быть ***ориентированны на тестирование*** компонентов и сложных комплексов программ. Для конфигурационного управления тестами, модификациями требований к комплексу программ и результатами испытаний, ***в базе данных конфигурации должны собираться сведения***, которые состоят из следующих ***основных частей***:

- данные об отказовых ситуациях, дефектах и ошибках, условиях их проявления и характеристиках обнаруживающих тестов, а также предложения на изменения функций программ, подлежащие анализу и селекции для выделения тех из них, для которых будут разрабатываться корректировки комплекса программ – ***база данных предлагаемых изменений требований***;
- разработанные тесты и корректировки программ отобранные группой конфигурационного управления для проведения изменений в очередной версии программного комплекса – ***база данных подготовленных и утвержденных корректировок требований и тестов***;
- откорректированные версии и набор изменений требований и тестов, выполненных в каждой из них – ***база данных утвержденных требований, тестов и реализаций версий программных комплек-***

сов и компонентов;

Эта ***информация в структурированных подсистемах базы данных управления конфигурацией*** должна быть защищена от случайных и преднамеренных искажений, путем организованного санкционирования, дублирования и контроля модификаций, истории их создания и изменения, в процессах жизненного цикла комплекса программ. Необходимо гарантировать сохранность версий изменений, с учетом их важности для результатов всего проекта. Особенно защищенным от искажений и разрушения, следует сохранять ***архив версий программных продуктов***, прошедших успешные испытания, утвержденные заказчиком, и скрепленные его подписью. Для устранения дефектов, реализации корректировок и ошибок при развитии новых версий целесообразно выделять рабочую копию предшествовавшей версии и архив накопленных изменений, обеспечивающих возможность «***отката»*** к предыдущей версии в случае разрушительных некорректных изменений в процессе разработки новой версии программного продукта.

Такая ***система обеспечения информацией процессов управления конфигурацией тестирования*** может быть структурирована на подсистемы в соответствии с адаптированной версией жизненного цикла конкретного комплекса программ. В соответствии с задачами специалистов проекта, на рис. 2.21 представлены ***основные подсистемы базы данных*** информационного обеспечения модификаций и тестирования, ориентированные на определенные процессы и компоненты комплексов программ. Для каждой подсистемы управления конфигурацией (левая колонка) целесообразно выделять достаточно автономную базу данных компонентов (правая колонка) с ограниченным доступом только для определенных категорий специалистов (см. таблицу 1). Эти подсистемы базы данных могут быть построены на стандартизированной основе СУБД проекта, взаимодействовать с аналогичными по структуре предшествующей и последующей базами данных жизненного цикла комплекса программ. Они должны накапливать и содержать основные компоненты и документы проекта на соответствующем уровне производства комплекса программ. Интерфейсы этого взаимодействия подсистем базы данных должны быть стандартизированы, по возможности ограничены по объему и доступности текущей и отчетной информации для других категорий специалистов. Для каждого сложного проекта комплекса программ целесообразно оформлять и утверждать ***Руководство и схему подсистем базы дан-***

ных, обеспечивающих управление конфигурацией комплекса программ, его требований и тестов, а также категории ответственных лиц за их поэтапную реализацию, контроль и сохранность информации всего проекта (см. **ISO 15846**).

Контроль корректности состояния и изменений требований и тестов должен обеспечивать регистрацию, оценку, рассмотрение и утверждение их состояния и изменений на протяжении жизненного цикла комплекса программ:

- обеспечивать целостность компонентов конфигурации и базовых версий комплекса, а также защиту их от некорректных изменений требований и тестов;
- гарантировать, что каждое изменение компонента конфигурации учтено в изменении идентификации конфигурации требований и тестов;
- зарегистрированы, утверждены и прослежены изменения в базовых версиях и компонентах конфигурации требований и тестов, находящихся под контролем;
- изменения требований к программному продукту прослежены вплоть до места их источника, а выполнение процессов жизненного цикла повторено с момента, начиная с которого изменения сказываются на выходных данных;
- при проведении работ по внесению изменений требований, модифицированы и обновлены документы жизненного цикла комплекса программ, на которые эти изменения могут влиять.

Руководство по документированию состояния конфигурационного управления требованиями и тестами конкретного сложного программного комплекса и его компонентов должно отражать:

- общую структуру комплекта документов на конфигурацию требований и тестов к программному продукту;
- номенклатуру и содержание каждого документа, отражающего требования и/или их изменения;
- требования к качеству, оформлению и обозначению каждого документа;
- регламент комплектования, корректировки и хранения документов;
- правила обращения, процессов корректировки и сопровождения документов, отражающих требования и тесты;

- графики подготовки, проверки, редактирования, согласования, утверждения и распространения документов.

Работы по созданию и применению документов конфигурационного управления должны гарантировать пользователям использование ***требований и тестов, только адекватных санкционированным версиям программных комплексов и компонентов***, так чтобы официальные документы могли быть получены только из архива. Каждая единица конфигурации должна быть идентифицирована, документирована и выпущена ее официальная версия до того, как осуществляется производство программных продуктов для пользователей. Цель работ по архивированию и получению документов – обеспечить получение документов, которые необходимы для возможности копирования, повторной генерации, повторного тестирования и модификации программного комплекса (см. рис. 2.21).

Методика оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке требований должна содержать рекомендации заказчикам, испытателям и пользователям по выявлению, регистрации и формализации условий проявления и содержания отказов и дефектов испытываемых и/или эксплуатируемых версий программного продукта. Эта методика должна включаться в состав эксплуатационной документации и передаваться ***каждому пользователю версии***. В методике следует стимулировать специалистов – пользователей, анализировать, подготавливать и представлять рекомендации заказчику и разработчикам по совершенствованию и развитию требований и тестов к основным функциям и характеристикам качества поставляемой версии программного продукта. Результаты анализа и предложения необходимо передавать для конфигурационного управления в унифицированной форме ***Отчетов специалистов о выявленных дефектах и предложениях по корректировке требований*** к программному продукту, которые должны содержать:

- подробное описание сценария функционирования и тестирования программного продукта, и исходных данных, при которых выявлен отказ или дефект;
- описание проявления отказовой ситуации или дефекта и документы результатов его регистрации;
- предположение о причине, вызвавшей проявление отказа или дефекта;
- предложение по корректировке требований к программному комплексу и его компонентам для устранения дефекта и/или совер-

шенствования функциональной пригодности и применения программного продукта.

Изменения, отобранные Советом конфигурационного управления проектом, переводятся из базы данных предварительных изменений требований в следующую подсистему базы данных. Кроме того, для каждой подготовленной корректировки следует регистрировать результаты ее рассмотрения Советом конфигурационного управления и утверждения на введение в новую версию программного продукта или для частных извещений пользователям. Отвергнутые корректировки возвращаются в базу данных предлагаемых изменений требований.

База данных выявленных отказов, дефектов и предложений по совершенствованию требований к функциям программ и тестов, а также результатов анализа предполагаемых корректировок версий должна содержать полные отчеты пользователей о выявленных дефектах и предложениях, а также дополнительно:

- результаты анализа причины и источника выявленного отказа или дефекта;
- предложения о возможных способах устранения отказа, дефекта или о реализации предложения по изменению требований, тестов и совершенствованию компонентов и комплекса программ;
- оценки сложности, трудоемкости, эффективности и срочности корректировки программ;
- оценки возможного влияния предлагаемых изменений на эксплуатацию версий программного продукта, имеющихся у пользователей.

Неожиданные или некорректные результаты тестов могут записываться в специальной подсистеме ведения отчетности по сбоям, обеспечивая формирование базы данных, используемой для отладки, устранения проблем и дальнейшего тестирования. Кроме того, аномалии, которые нельзя идентифицировать как сбои, также могут фиксироваться в системе ведения отчетности по сбоям. В любом случае, документирование таких аномалий снижает риски процесса тестирования и помогает решать вопросы повышения надежности тестируемой системы. Отчеты по тестам могут являться входом для процесса управления изменениями и генерации запросов на изменения при отслеживании дефектов. Все корректировки требований, утвержденные для введения в очередную версию, следует регистрировать в следую-

щей подсистеме базы данных. Для каждого изменения должны документироваться содержательная аннотация, а также общие характеристики и достигнутое на испытаниях качество очередной версии программного продукта.

Отчеты о дефектах и корректирующих действиях требований и тестов, связанные с процессами жизненного цикла комплексов программ, целесообразно фиксировать в отдельной ***подсистеме*** конфигурационного управления:

- должно быть подготовлено сообщение о каждом дефекте требований и его реализации, отказовые ситуации или аномальное поведение программного компонента и/или комплекса, а также предпринятые корректирующие действия;
- отчеты о дефектах и отказовых ситуациях должны предусматривать идентификацию затрагиваемых компонентов и требований, состояние сообщений о дефектах, утверждение и закрытие сообщений о дефектах после корректировок;
- сообщения о дефектах и отказовых ситуациях, для которых требуются корректирующие действия требований в программном продукте или выходных данных процессов жизненного цикла, должны активизировать работы по выполнению и контролю изменений требований и тестов.

Тестировщики должны составлять ***сообщения о дефектах и изменениях требований и тестов,*** чтобы описать каждый дефект, обнаруженный в программном комплексе, находящийся под контролем конфигурации и каждую работу по тестированию модификации требований, необходимых по контракту или описанных в плане разработки. Система производства комплекса должна быть закрытым циклом, гарантирующим, что все обнаруженные дефекты и отказовые ситуации ***вводятся в систему регистрации***, необходимые модификации требований инициируются, состояния корректирующих действий отслеживаются, и сообщения об изменениях требований сопровождаются в течение срока действия контракта. Каждый дефект или модификация требований должны быть классифицированы по категориям и приоритетам, тесты и корректирующие действия должны быть оценены, чтобы определить были ли дефекты устранены, а изменения компонентов и комплекса программ были правильно выполнены без внесения дополнительных дефектов.

Подсистема базы данных подготовленных и утвержденных результатов корректировок требований, а также реализованных

изменений и обобщенных характеристик качества новой версии программного продукта должна содержать:

- причина корректировки требований комплекса программ и базы данных (отказ, ошибка, дефект, совершенствование);
- содержание изменений требований программ и базы данных, а также документации на версию программного продукта;
- результаты тестирования версии программного продукта с предполагаемыми изменениями требований;
- результаты квалификационных испытаний и обобщенные характеристики качества версии программного продукта после внесения изменений требований;
- решение по распространению пользователям проведенной модификации требований и/или версии программного продукта;
- адрес хранения корректировок, документов и квалификационных тестов новой версии программного продукта.

Конфигурационная база реализованных изменений требований и тестов должна быть установлена соглашением разработчиков с заказчиком, и использоваться для официального контроля за конфигурацией требований и их реализацией. После первоначального выпуска документов на конфигурацию версии программного продукта все изменения следует контролировать. Влияние изменения на продукт, требования заказчика и подвергнутая влиянию конфигурационная база должны определять степень формальности, соблюдаемой при работе с изменениями, и могут стать основой системы классификации, используемой для распределения изменений требований по категориям и приоритетам. Разрушение сведений, о выполненных изменениях программного продукта, может приводить к большими затратам на их восстановление. Поэтому база данных изменений требований, тестов и их последствий должна дублироваться и поддерживаться методами и средствами сопровождения, аналогичными, применяемыми для основной документации, тестов и текстов комплекса программ.

Отчеты о состоянии конфигурации требований, компонентах и комплексе программ должна содержать информацию о всех идентификациях конфигурации, всех отклонениях от установленных требований к конфигурации подсистем баз данных. Проверки конфигурации требований следует проводить до принятия конфигурационной базы программного продукта заказчиком с целью гарантии того, что продукция соответствует контрактным или установленным тре-

бованиям, и что она точно отражена в документах на конфигурацию продукта. Такие проверки могут требоваться для официальной приемки конфигурации версии программного продукта.

Особое значение при сопровождении требований и тестов имеет ***документация на реализованные корректировки и тесты***, с помощью которых проверялась корректность версий компонентов и комплекса в целом. Эта документация должна позволять ***восстанавливать историю разработки*** и проверки каждого изменения требований для любого компонента. Разработка и тестирование изменений может опережать их документальное оформление. В течение этого времени возможны отдельные уточнения изменений требований в версии. В результате документация должна непрерывно ***«догонять»*** реальное состояние программного комплекса на базе данных конфигурации. Для упорядочения этого процесса стандартами установлена возможность оперативного выпуска ***предварительных извещений на частные изменения требований и их реализацию***. Эти извещения регистрируются как временные и погашаются при полном оформлении документации на версию программного продукта и все реализованные изменения.

Перечисленные ***документы в подсистемах базы данных*** реально могут структурироваться на более мелкие фрагменты. Они взаимосвязаны и сведения об изменениях по мере обработки должны переходить последовательно из одной части базы данных в другую. Первичные сообщения от испытателей или пользователей об отказах, дефектах и ошибках следует регистрировать в специализированных таблицах, отражающих их формализованные параметры, а также на содержательном уровне вместе ***с тестовыми данными и ситуациями***, при которых зарегистрирован дефект. Также на содержательном уровне или в виде спецификации требований необходимо регистрировать в базе данных предложения по совершенствованию качества комплекса программ. По результатам анализа и тестирования от группы конфигурационного управления эти данные получают признаки отвергнутых изменений требований или подлежащих дальнейшей проработке. Отвергнутые изменения требований через некоторое время могут исключаться из базы данных предлагаемых изменений.

Для эффективной реализации процессы конфигурационного управления требованиями и тестами сложных комплексов программ должны быть поддержаны ***службой тиражирования, архивирования и гарантированного хранения*** всей необходимой информации о документах, их корректировках, версиях, авторах и их правах на изменения.

Эта ***служба архива*** должна обеспечивать поставку пользователям только утвержденных копий версий программного продукта и/или их компонентов с полным, адекватным комплектом эксплуатационных документов, а также извещений на частные изменения конкретных, ранее приобретенных версий. Для гарантированного сохранения состояния и результатов модификаций требований к программам и обеспечения возможности их анализа на любой стадии проекта, а также ***отката по истории*** выполненных корректировок требований и компонентов, следует организовать четкую ***систему хранения и копирования подлинников, дубликатов и копий требований и тестов*** программного продукта и документов.

Разработчики должны создавать периодические отчеты о состоянии конфигурации и идентификации реализации всех версий требований и тестов, которые были помещены под управление конфигурацией в архиве. Эти отчеты для заказчика должны поддерживаться в течение срока действия контракта. Они должны включать информацию о текущем состоянии базовой версии требований, официальной выпускаемой версии каждого компонента, и информацию об истории изменений требований системы, компонентов и тестов с момента их помещения под контроль конфигурации, а также о состоянии сообщений о дефектах и изменениях.

Модификация, учет и тиражирование версий сложного программного комплекса и компонентов может ***требовать больших затрат***. Поэтому при выпуске каждой новой базовой версии разработчики стремятся обеспечить преемственность ее функций, компонентов и документов с предыдущими версиями, а также рассматривается возможность и подготавливается решение для прекращения модификаций некоторой устаревшей версии или ее конкретных компонентов. Ряд процедур в стандартах сконструирован так, что возможна их ***адаптация*** в соответствии с характеристиками конкретных проектов и внешней среды программных продуктов пользователей. Для реализации на практике процедур и планов управления конфигурацией требований к программным продуктам и тестам необходимы организационные мероприятия, гарантирующие участникам проектов определенную ***культуру, дисциплину разработки и выполнения их модификаций***. Такая организационная система должна обеспечивать специалистам разной квалификации и роли в проекте, возможность взаимодействия при решении требуемых комплексных задач, для на-

копления, хранения и обмена упорядоченной информацией о состоянии и изменениях требований, тестов и компонентов комплекса программ.

Наиболее сложной задачей установки реальных ***ориентиров затрат*** для заказчика является оценка размера программного продукта и его изменений. Нереальные ожидания, основанные на неточных оценках требуемых ресурсов, представляют собой одну из ***частых причин провала проектов*** вследствие превышения ограничений доступных ресурсов. Так как, величина и достоверность определения размера возможных модификаций комплекса программ, составляют ключевой фактор последующего анализа влияния ограничений ресурсов, то целесообразно применять несколько доступных методов для его оценивания. Конкретизация функций, структуры, состава компонентов и комплекса в целом, позволяет более достоверно определять размеры возможных модификаций групп программ и, суммируя их, оценивать размер ресурсов, необходимых для программного комплекса. Особенно сильно на снижение суммарных затрат влияет использование готовых компонентов и документов из предшествующих разработок. При анализе аналогов могут быть выделены компоненты, пригодные для повторного применения в новом проекте или в версии программного комплекса. Это позволяет оценить возможную долю использования готовых компонентов и тем самым определить эффективный размер комплекса программ и документов, подлежащий модификациям и размер информации ***в базе данных управления конфигурацией*** (см. лекцию 1.6). На базе всего комплекса использованных тестов должна создаваться и документироваться для каждой версии программного продукта, ***эталонная тестовая (контрольная) задача и контрольные результаты ее решения***. Эти документы следует оформлять в соответствии со стандартами, тиражировать и передавать пользователям вместе с версией программного продукта и остальными эксплуатационными документами.

Проводимый в учебнике анализ ориентирован на сложные комплексы программ высокого качества. Многие проекты являются более простыми, для этого целесообразно проводить адаптацию и формировать ***практическую рабочую версию Руководства управления конфигурацией*** при тестировании конкретного проекта, которая должна регламентировать работы специалистов. Известные функции, потенциальные пользователи и концепции существующих версий программного продукта позволяют прогнозировать направления совершенствова-

ния и уменьшения модификаций для нового проекта или версии, имеющих прототип.

Эффективность процессов разработки и совершенствования версий программного продукта в значительной степени определяется их организацией и информационным обеспечением баз данных. ***Регламентированное конфигурационное управление*** повышает качество проекта, сокращает непроизводительные затраты труда и времени на каждом этапе жизненного цикла комплекса программ, обеспечивает унифицированный, гибкий процесс модификации и развития множества версий. Выбор технологии разработки программ и СУБД в значительной степени зависит от перспективы последующего сопровождения и совершенствования комплекса программ. Контроль среды поддержки жизненного цикла состоит в обеспечении гарантий того, что инструментальные средства, используемые для создания и модификации программ, – идентифицируются, контролируются и могут быть получены из соответствующих источников. Для этого должна оцениваться возможная длительность сопровождения и тираж различных версий у пользователей, которые следует учитывать при выборе технологии, баз данных и инструментария для разработки компонентов и комплекса программ в целом.

Для конкретного проекта должны быть определены и зафиксированы ***правила управление конфигурацией, оценки и утверждения интегрального риска версий программного продукта***, применения административных и технологических процедур их мониторинга на всем протяжении жизненного цикла для:

- идентификации, определения и регистрации конфигурации и изменений программного продукта в системе;
- управления конфигурацией, модификацией и выпуском версий программных продуктов при сокращении интегрального риска;
- фиксирования конфигурации и сообщений о состоянии версий программного продукта и его компонентов после утверждения интегрального риска;
- управления и контролирования хранения, обращения и поставок версий программных продуктов после реализации контрмер и достижения допустимого риска.

Любое изменение программного продукта может ***потребовать пересмотра соглашений*** с заказчиком о функциональности, качестве, сроках и бюджете, присущих производству системы. Группы управ-

ления изменениями часто становятся частью такого процесса и организуют форум для открытых дискуссий и поиска компромиссов с заказчиком по поводу ранее принятых решений. Необходимо быть внимательным к организационным, функциональным, технологическим и проектным изменениям и принимать в расчет любые дополнительные знания о проекте, которые оказывают ***воздействие на риски программных продуктов***.

Управление сертификацией программных продуктов

Для сложных программных продуктов с особенно высокими требованиями к качеству типовое проектирование, производство, тестирование и испытания не всегда могут гарантировать полную реализацию всех требований и эталонов заказчиков и пользователей комплексов программ. Для этого необходимы экономические и моральные стимулы, а также ***воля*** руководителей, организация специалистов – исполнителей, методы и технология ***для сертификации создаваемых программ независимыми предприятиями***. Радикальное повышение качества сложных программных продуктов и ***обеспечение их конкурентоспособности*** на мировом рынке возможно только на базе внедрения современных стандартизированных технологий и систем качества, сертифицирующих, поддерживающих и контролирующих весь жизненный цикл программного продукта.

Характеристики качества продукта или процесса зависят от того, для какой ***цели***, для какого ***потребителя*** и для каких ***условий*** внешней среды делается их оценка. Один и тот же объект может иметь несколько различных представлений и оценок качества, произведенных для различных целей и при разных условиях применения. Различия фактических и требуемых показателей качества продуктов или процессов квалифицируются как дефекты или ошибки и являются первичными стимулами для реализации решений по изменению определяемых значений качества.

Объективное повышение сложности функций, реализуемых программами, непосредственно приводит к увеличению их размеров и трудоемкости создания. Соответственно росту сложности программ ***возрастает количество выявляемых и остающихся в них дефектов и ошибок***, что отражается на качестве функционирования. По мере увеличения сложности задач, решаемых программами, ошибки могут угрожать катастрофами в системах, выполняющих ***критиче-***

ские функции управления крупными, дорогими и особенно важными объектами или процессами. Разработка и сопровождение сложных программных продуктов на базе современных технологий позволяет предупреждать и устранять наиболее опасные системные и алгоритмические ошибки на ранних стадиях проектирования, а также использовать неоднократно проверенные в других проектах программные и информационные компоненты высокого качества. Однако, этого может быть не достаточно для гарантирования и удостоверения особенно высокого качества ряда критических программных продуктов.

Сложность анализируемых объектов – комплексов программ и психологическая самоуверенность ряда программистов в собственной ***«непогрешимости»***, зачастую приводят к тому, что реальные характеристики качества функционирования программных продуктов остаются неизвестными не только для заказчиков и пользователей, но также для самих разработчиков. Проекты оказываются неудачными или даже терпят полный провал из-за недостаточной компетентности привлекаемых разработчиков, их ***неадекватного «оптимизма»***, а также вследствие слабого использования современных методов, технологий и стандартов, обеспечивающих требуемое высокое качество продуктов. Отсутствие четкого декларирования в документах понятий и требуемых значений характеристик качества вызывает конфликты между заказчиками-пользователями и разработчиками-поставщиками вследствие неполноты и разной трактовки одних и тех же характеристик.

Возможности и широта применения сертификации при производстве сложных комплексов программ существенно зависит от ***методологий и стилей*** организации работы специалистов – разработчиков. Эти методологии различаются сферами применения, методами достижения высокого качества комплексов программ, психологическими характеристиками участвующих специалистов и организацией их деятельности в реальном времени. В соответствии со стандартами, ***обеспечение качества*** – это «совокупность планируемых и систематически проводимых мероприятий, необходимых для уверенности в том, что продукция или процессы удовлетворяют определенным требованиям потребителей к качеству». Для реализации этого положения предназначены технологические ***системы обеспечения качества***, каждая из которых включает: «совокупность организационной структуры, процедур, процессов и ресурсов, обеспечивающую

ответственность руководства за качество процессов производства и/или продукции».

Потребителей – заказчиков интересует, прежде всего, ***качество готового поставляемого программного продукта*** и обычно не очень беспокоит, как оно достигнуто. Для ряда критических программных комплексов и систем это качество должно быть ***ответственно удостоверено и гарантировано*** компетентными, независимыми организациями путем достаточно широких, дополнительных регламентированных испытаний. Гарантирование качества продукции возможно посредством сертификационных испытаний реализуемых ***процессов производства*** комплексов программ и/или путем испытаний их результатов – ***программных продуктов***. ***Сертификация*** – процедура подтверждения соответствия требованиям, посредством которой независимое от изготовителя и потребителя предприятие ***юридически удостоверяет*** в письменной форме, что состояние продукции и/или производства и системы менеджмента качества способно обеспечить стабильность характеристик изготовляемой продукции и соответствует установленным заказчиком требованиям и стандартам (см. Приложение 1). Качество при проектировании и производстве программных продуктов можно обеспечивать ***двумя методами*** – рис. 2.22.

- посредством применения регламентированных технологий и систем обеспечения качества процессов проектирования и производства, предотвращающих дефекты, контролирующих и гарантирующих высокое качество продуктов в процессе их создания;
- путем использования заключительного контроля и испытаний готовых поставляемых продуктов, и исключения из поставки или направлением на доработку изделий, не соответствующих требуемым показателям качества.

Первый метод обеспечивает и контролирует высокое качество выполнения всего процесса проектирования и производства, и тем самым минимум экономических потерь от брака, что рентабельно при создании сложных систем. Многие руководители осознали, что для создания современных прикладных высококачественных информационных систем необходимы не менее ***качественные технологии*** их производства. При этом качество тех и других должно удостоверяться регламентированными поэтапными и заключительными испытаниями.

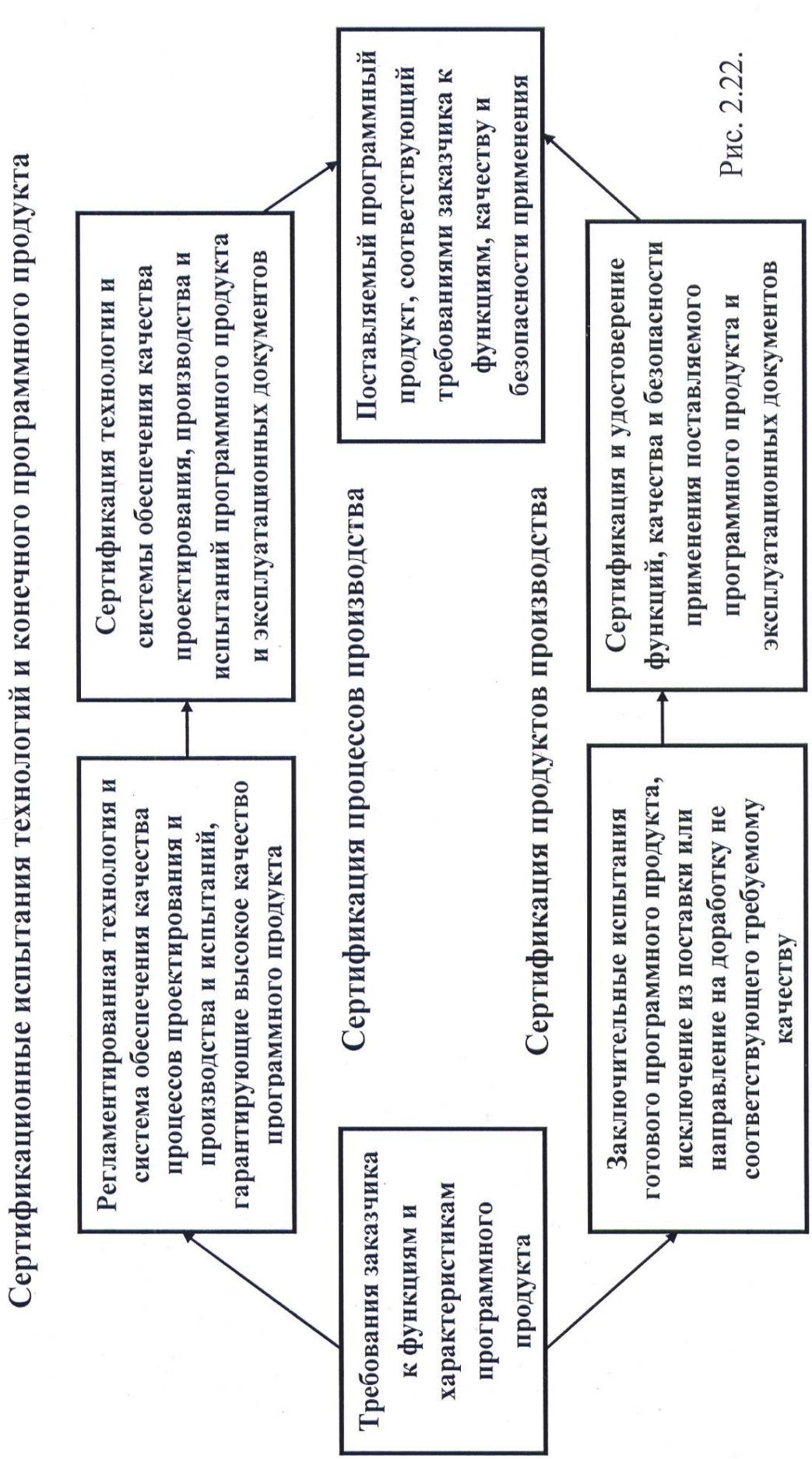

Сертификационные испытания технологий и конечного программного продукта
Регламентированная технология и система обеспечения качества процессов проектирования и производства и испытаний, гарантирующие высокое качество программного продукта
Сертификация технологии и системы обеспечения качества проектирования, производства и испытаний программного продукта и эксплуатационных документов
Сертификация процессов производства
Требования заказчика к функциям и характеристикам программного продукта
Поставляемый программный продукт, соответствующий требованиями заказчика к функциям, качеству и безопасности применения
Сертификация продуктов производства
Заключительные испытания готового программного продукта, исключение из поставки или направление на доработку не соответствующего требуемому качеству
Сертификация и удостоверение функций, качества и безопасности применения поставляемого программного продукта и эксплуатационных документов

Рис. 2.22.

Качество технологии не всегда можно достоверно контролировать и гарантировать ее соблюдение специалистами, однако оно отражается на качестве продукции, а, следовательно, при этом может оставаться не полностью определенным достигаемое качество программных продуктов. Результаты испытаний ***процессов производства*** трудно измерять количественными критериями, и обычно характеризуют рядом требований к качественному выполнению стандартизированных производственных процессов. Они оцениваются набором свойств, непосредственно отражающихся на характеристиках качества конечного программного продукта, однако при этом нет гарантии адекватного и однозначного на них влияния. Этот метод может приводить при производстве к ***неопределенности качества*** компонентов и комплексов программ в целом и к значительным экономическим потерям за счет затрат на создание не пригодного к использованию брака, что может быть очень дорого для сложных систем.

Второй метод акцентирован на анализ и контроль качества готового поставляемого программного продукта, которое удостоверяется при сертификационных испытаниях. Достижение при производстве необходимого качества продукции за счет только выходного контроля, при отсутствии или недостатках системы обеспечения качества в технологическом процессе разработки, может приводить к длительному итерационному процессу массовых доработок и повторных испытаний. При сертификационных испытаниях качества готового ***программного продукта*** могут использоваться стандартизированные количественные и качественные критерии и характеристики, которые непосредственно отражают функции и свойства продукции, интересующие заказчика и потребителей, их можно измерить и достоверно установить.

Таким образом, ***задача обеспечения и удостоверения высокого качества*** сложных программных продуктов сводится ***к испытаниям*** технологий процессов проектирования и производства программных средств, поддержанных системой качества, и конечного продукта, созданного на базе таких технологий. Соответственно можно выделить ***два вида сертификационных испытаний***: ***технологий*** обеспечения жизненного цикла программных средств, поддержанных регламентированными системами качества и/или готового программного ***продукта*** с полным комплектом эксплуатационной документации (см. рис. 2.22).

Взаимосвязь качества разработанных комплексов программ с качеством технологии их создания и с затратами на производство становится особенно существенной при необходимости получения ***критического конечного продукта*** с особенно высокими значениями характеристик качества. Затраты на производство обычно резко возрастают, когда требуемые показатели качества приближаются к пределу, достижимому при данной технологии и уровне автоматизации процессов проектирования и производства. В этих случаях для обеспечения высокого качества необходима совместная ***сертификация технологий и системы обеспечения качества*** их проектирования, производства, тестирования и сопровождения, а ***также сертификация готового программного продукта***. Этот вид комплексной сертификации обеспечивает контроль реализации требований алгоритмической и функциональной корректности программного продукта, что особенно важно в программных комплексах для ***обеспечения безопасности критических систем***, а также сокращения случайных дефектов и ошибок программ. Сертификация технологических процессов относительно слабо связана с конкретной функциональностью продукта и должна обеспечивать, в основном, его конструктивную безопасность – минимум рисков, дефектов и технических ошибок. Зачастую это обстоятельство не учитывается в важных критических системах, в которых может оставаться дефект или ошибка в каждой тысяче строк программного кода, которые способны ***резко снижать безопасность*** сложных программных продуктов.

Наиболее полно в России стандартизирована ***сертификация производства товарной продукции*** различных видов. Она поддержана стандартами: Временный порядок сертификации производств с учетом требований **ГОСТ Р ИСО 9001:2001**; **ГОСТ Р 40.003: 2005** и **ГОСТ Р ИСО 19011: 2003** (см. Приложение 1). Их концепции определены в ***Системе менеджмента качества производства***. При этом ***сертификация производства*** определена как ***процедура подтверждения соответствия***, посредством которой независимая от изготовителя (продавца, исполнителя) и потребителя (покупателя) организация удостоверяет в письменной форме, что состояние производства (системы менеджмента качества производства) способно обеспечить стабильность характеристик изготовляемой продукции и соответствует требованиям **ГОСТ Р ИСО 9001**. К работе по сертификации привлекаются эксперты (аудиторы) по сертификации произ-

водств, зарегистрированные в Регистре системы сертификации персонала – ***сертифицированные специалисты***. Область сертификации производства определяет заказчик по согласованию с председателем комиссии органа по сертификации продукции.

На практике акцент, распределение ресурсов и усилий на эти два вида сертификации зависят от особенностей характеристик комплекса программ, квалификации коллектива специалистов – разработчиков, требований заказчика – потребителей и наличия сертификационной лаборатории соответствующей тематической квалификации. Для этого организация и процессы сертификации должны специализироваться на определенные виды сертификации и на определенные классы программных продуктов, предусматривать соответствующие технологические работы и документы, обеспечивающие создание продукта требуемого качества. В общем случае ***специализация сертифицирующих коллективов*** для испытаний технологий и систем качества может быть более широкой и универсальной, чем для сертификации конкретных программных продуктов, которые описываются более определенными и точными критериями качества. Сертифицированные версии программного продукта и технологические результаты процессов сертификации целесообразно сохранять в специальной ***подсистеме базы данных конфигурационного управления*** (см. рис. 2.21). Это олжно позволять отслеживать причины и характеристики изменений и дефектов по всем этапам тестирования и испытаний программного продукта.

Кроме сертификации технологических процессов и готовых программных продуктов для эффективного их производства и применения, важное значение имеет ***сертификация квалификации специалистов – аудиторов***, реализующих эти процессы. Для этого необходимо их обучение и аттестация на допуск к участию в таких работах, требующих определенных уровней профессиональной квалификации. Они должны освоить и знать методологию программной инженерии, методы тестирования и документирования программных средств, а также основы сертификации производственных процессов и продуктов. Методы и процессы программной инженерии, должны включаться в Программы комплексного обучения для обеспечения необходимой профессиональной квалификации руководителей и специалистов, освоения и применения дисциплин регламентированной деятельности крупных коллективов.

Приложение 1

Международные и государственные стандарты, регламентирующие требования и тестирование компонентов и комплексов программ

1. **CMMI – Capability Maturity Model Integration for Product and Process Development –** Интегрированная модель оценивания зрелости продуктов и процессов разработки программных средств.
2. **ISO 19759:2005. SWEBOK.** Свод знаний о программной инженерии.
3. **ISO 15288:2002.** Системная инженерия. Процессы жизненного цикла систем.
4. **ISO 19760:2003.** – Системная инженерия. Руководство по применению стандарта ISO 15288.
5. **ISO 12207:1995.** (ГОСТ Р – 1999). ИТ. Процессы жизненного цикла программных средств.
6. **ISO 12207:1995.** – ИТ. Процессы жизненного цикла программных средств. **Изменения 1 и 2:2002-2004**.
7. **ISO 15271:1998.** (ГОСТ Р – 2002). ИТ. Руководство по применению ISO 12207.
8. **ISO 16326:1999.** (ГОСТ Р – 2002). ИТ. Руководство по применению ISO 12207 при административном управлении проектами.
9. **ISO 15504 – 1-5: 2003-2006.** ИТ. Аттестация процессов. Ч.1. Концепция и словарь. Ч.2. Подготовка к аттестации. Ч.3. Руководство по проведению аттестации. Ч.4. Руководство для пользователей по усовершенствованию процессов и определению зрелости процессов. Ч.5. Образец модели аттестации процессов.
10. **ГОСТ Р 51904 – 2002**. Программное обеспечение встроенных систем. Общие требования к разработке и документированию.
11. **ISO 9000:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Основы и словарь.
12. **ISO 9001:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Требования.
13. **ISO 9004:2000.** (ГОСТ Р – 2001). Система менеджмента (административного управления) качества. Руководство по улучшению деятельности.

14. **ISO 90003:2004** – Руководство по организации применения стандарта **ISO 9001:2000** для программных средств.
15. **ISO 10005: 1995** - Административное управление качеством. Руководящие указания по программам качества.
16. **ISO 10006: 1997** - Руководство по качеству при управлении проектом.
17. **ISO 10007: 1995** - Административное управление качеством. Руководящие указания при управлении конфигурацией.
18. **ISO 10011-1-3: 1990.** Руководящие положения по проверке систем качества. Ч.1. Проверка. Ч.2. Квалификационные критерии для инспекторов-аудиторов систем качества. Ч.3. Управление программами проверок.
19. **ISO 12182:1998.** (ГОСТ Р–2002). ИТ. Классификация программных средств.
20. **ISO 9126:1991.** (ГОСТ – 1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.
21. **ISO 14598-1-6:1998-2000.** Оценивание программного продукта. Ч.1. Общий обзор. Ч.2. Планирование и управление. Ч.3. Процессы для разработчиков. Ч.4. Процессы для покупателей. Ч.5. Процессы для оценщиков. Ч.6. Документирование и оценивание модулей.
22. **ISO 9126-1-4: 2002**. ИТ. ТО. Качество программных средств: Ч.1. Модель качества. Ч.2. Внешние метрики. Ч. 3. Внутренние метрики. Ч.4. Метрики качества в использовании.
23. **ISO 25000:2005** ТО. – Руководство для применения новой серии стандартов по качеству программных средств на базе обобщения стандартов ISO 9126:1-4: 2002 и ISO 14598:1-6:1998-2000.
24. **ISO 15939: 2002** – Процесс измерения программных средств.
25. **IEC 61508:1-6: 1998-2000.** Функциональная безопасность электрических / электронных и программируемых электронных систем. Часть 3. Требования к программному обеспечению. Часть 6. Руководство по применению стандартов IEC 61508-2 и IEC 61508-3.
26. **ISO 15408 -1-3. 1999.** (ГОСТ Р – 2002). Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Ч.1. Введение и общая модель. Ч.2. Требования к функциональной безопасности. Ч.3. Требования к доверию безопасности.

27. **ISO 13335 - 1-5. 1996-1998.** ИТ. ТО. Руководство по управлению безопасностью. Ч.1. Концепция и модели обеспечения безопасности информационных технологий. Ч.2. Планирование и управление безопасностью информационных технологий. Ч.3. Техника управления безопасностью ИТ. Ч.4. Селекция (выбор) средств обеспечения безопасности. Ч.5. Безопасность внешних связей.
28. **ISO 10181: 1-7. BOC. 1996-1998.** Структура работ по безопасности в открытых системах. Ч.1. Обзор. Ч.2. Структура работ по аутентификации. Ч.3. Структура работ по управлению доступом. Ч.4. Структура работ по безотказности. Ч.5. Структура работ по конфиденциальности. Ч.6. Структура работ по обеспечению целостности. Ч.7. Структура работ по проведению аудита на безопасность.
29. **ISO 16085:2005.** SE. Процессы жизненного цикла программных средств. Управление рисками.
30. **ISO 14252:1996** (IEEE 1003.0). Руководство по функциональной среде открытых систем POSIX.
31. **ISO 9945-1:1990** (IEEE 1003.1). ИТ. Интерфейсы переносимых операционных систем. Ч.1. Интерфейсы систем прикладных программ (язык Си).
32. **ISO 9945-2:1992** (IEEE 1003.2). ИТ. Интерфейсы переносимых операционных систем. Часть 2. Команды управления и сервисные программы.
33. **ISO 13210:1994.** ИТ. Методы тестирования для измерения соответствия стандартам POSIX.
34. **ISO 14756: 1999.** ИТ. Измерение и оценивание производительности программных средств компьютерных вычислительных систем.
35. **ISO 12119:1994.** (ГОСТ Р – 2000 г). ИТ. Требования к качеству и тестирование.
36. **ANSI/IEEE 829 - 1983**. Документация при тестировании программ.
37. **ANSI/IEEE 1008 - 1986**. Тестирование программных модулей и компонентов программных средств.
38. **ANSI/IEEE 1012 - 1986**. Планирование верификации и подтверждения достоверности качества (валидации) программных средств.

39. **ISO 14764: 1999.** (ГОСТ Р – 2002). ИТ. Сопровождение программных средств.
40. **ISO 15846:1998.** ТО. Процессы жизненного цикла программных средств. Конфигурационное управление программными средствами.
41. **ISO 15910:1999.** (ГОСТ Р – 2002) ИТ. Пользовательская документация программных средств.
42. **ISO 18019:2004** ИТ. Руководство по разработке пользовательской документации на прикладные программные средства для офисов, бизнеса и профессиональных применений.
43. **ISO 6592:2000.** ОИ. Руководство по документации для вычислительных систем.
44. **ISO 9294:1990.** (ГОСТ–1993 г). ТО. ИТ. Руководство по управлению документированием программного обеспечения.
45. **ГОСТ 34.602-89**. ИТ. Техническое задание на создание автоматизированных систем.
46. **ГОСТ 34.603-92**. ИТ. Виды испытаний автоматизированных систем.
47. **РД 50-34.698-90.** Методические указания. Информационная технология. Автоматизированные системы. Требования к содержанию документов.
48. **ГОСТ Р 51901-2002.** Управление надежностью. Анализ риска технологических систем.
49. **DO-178 B-1995.** Соглашение по сертификации бортовых систем и оборудования в части программного обеспечения.
50. **ISO 14102:1995.** Оценка и выбор CASE- средств.
51. **ISO 14471:1995.** Руководство по адаптации CASE- средств.
52. **ISO 14143: 1-5: 1998 – 2004.** ИТ. Измерение программных средств. Измерение функционального размера. Ч.1. 1998. Определение концепции. Ч.2. 2002. Оценивание соответствия методов измерения размера программных средств стандарту ISO 14143:1:1998. Ч.3. 2003. Верификация методов измерения функционального размера. Ч.4. 2002. Эталонная модель. Ч.5. 2004. Определение функциональных доменов для использования при измерении функционального размера.
53. **ISO 20926:2003.** Руководство по практическому методу измерения функционального размера программных средств.
54. **ISO 20968:2002.** Руководство по расчетам на основе анализа функциональных точек – Марк II.

Приложение 2

Основы построения и применения графов потоков управления и потоков данных программных модулей и компонентов для тестирования

Графы – способ структурировать мышление, полезный для получения и анализа корректных тестов программных компонентов и модулей. ***Графы*** являются набором объектов, отношений между этими объектами и спецификаций, указывающих, какие объекты связаны и каким образом [Бе, Евст]. Далее под объектами подразумеваются программные модули и компоненты, элементы их структуры и свойств. Граф – это набор узлов, имен узлов, весов узлов, связей, имен связей, весов связей и отношений между узлами (см. рис. 2.5). [Бе].

Узлы – основные объекты в графах изображаются кружочками. Задача тестирования – убедиться, что граф имеет все требуемые, необходимые узлы и не более того. Каждый узел должен иметь ***уникальное имя***. Если объекты является файлами, имена узлов могут быть именами файлов; если объекты – это программы или операторы программ, имена узлов могут быть именами программ или метками операторов соответственно. Узлы могут иметь свойства: ***вес узла*** – состояние элемента программы, значение переменной, функция, которая описывает, какая из нескольких величин может быть использована для вычисления чего-либо, или имя другого объекта.

Связи – стрелка, линия или дуга, которая соединяет узлы и используется для иллюстрации конкретного отношения между этими узлами. Если в программе определено только одно отношение, можно не надписывать имя каждой связи, чтобы обозначить это отношение. Если между объектами может быть несколько отношений, можно помечать связи именами заданных отношений. Параллельные связи используются, если между двумя узлами существуют несколько отношений. Каждая связь должна иметь ***уникальное имя***. Связи могут обладать свойствами – ***весом связи.*** Если узлы обозначают шаги обработки, а связи показывают последовательность этих шагов, то весом связи может быть время выполнения программы по данному пути или вероятность выполнения именно этой части маршрута. Свойства графа, напрямую зависят от веса связей. ***Задача тестирования*** – убедиться, что связи с весом имеют требуемый вес.

Направленная связь изображается стрелкой и используются для обозначения несимметричных отношений между узлами лишь в заданном направлении. ***Ненаправленная связь*** соответствует ***симметричному*** отношению, двустороннему отношению. Симметричные отношения изображаются двусторонними стрелками, стрелки на концах линий обычно опускаются. ***Направленный граф,*** в котором все связи направленные. Большинство графов в тестировании программных модулей и компонентов – направленные.

Выходной узел – без исходящих связей, ни одна стрелка на них не начинается.

Путь – маршрут от входного узла к выходному узлу – не обязательно является реализуемым путем при тестировании текста программы. В тестировании исполнения программ рассматриваются пути, описывающие поведение программных компонентов. ***Проходимый путь*** – для которого существует такой набор входных значений, что, используя их, программа может пройти до конца по этому маршруту. ***Непроходимый путь***, который невозможно воспроизвести в исполняемой программе, какой бы набор входных значений не вводился. Если отдельно не оговаривается, то под термином ***«путь – маршрут»*** подразумевается путь от входа до выхода. Существует два способа назвать путь: по именам узлов вдоль этого пути или по именам связей вдоль маршрута. ***Длину пути можно измерить*** числом узлов вдоль этого пути или количеством связей вдоль маршрута.

Цикл – это маршрут, в котором как минимум один узел встречается больше одного раза, или в имени которого больше одного раза встречается имя связи, в зависимости от того, как строится имя пути – перечислением узлов или перечислением связей (см. рис. 2.5 узлы 6 – 8). ***Ациклический путь***, в котором нет циклов, в его имени не повторяются имена узлов (связей). ***Число повторений цикла*** – зависит от значения переменной управления циклом, если она существует. ***Детерминированный цикл,*** – число итераций которого, известно до того, как начнется выполнение цикла. ***Недетерминированный цикл***, – число итераций которого неизвестно до того, как начнется выполнение цикла, или цикл, число итераций которого определяется, или изменяется внутри цикла по ходу его выполнения.

Узел управления циклом, с двумя и больше исходящими связями: для одной связи цикл будет выполняться, а для другой не будет. ***Узел выхода из цикла*** представляет предикат, по крайней мере, с одним значением, вызывающим отмену выполнения цикла. Цикл может иметь более одного узла выхода и более одного узла входа.

Предикат управления циклом, значение которого определяет, будет цикл выполняться или нет. ***Переменная управления циклом*** в предикате управления циклом, значение которой влияет на значение предиката управления циклом – будет цикл выполняться или нет. ***Цикл с предусловием***, – в котором предикат управления циклом вычисляется до того как выполняется какая-либо обработка внутри цикла. ***Цикл с постусловием***, в котором предикат управления циклом вычисляется после обработки. В циклах с постусловием обработка выполняется, по меньшей мере, один раз.

Вложенные циклы – когда один цикл полностью содержится в другом (см. рис. 2.5 узлы 9 – 10 вложены в 7 – 11). Вложенные циклы целесообразно сначала протестировать как индивидуальные. Для внутреннего цикла задается типичное значение и тестируется внешний цикл для случаев критических значений, затем изменяется процедура на противоположную, задается для внешнего цикла типичное значение и исполняется внутренний цикл через критические тестовые варианты. Эти тесты должны выявить большинство ошибок, обычно попадающихся в одиночных, не вложенных циклах. Проблема с вложенными циклами возника-

ет, когда два или более вложенных цикла достигают критических значений одновременно, например, нулевых, максимальных, и так далее. Их тестовые варианты проектируются путем перебора комбинаций критических значений параметров циклов.

Ошибки циклов обычно обнаруживаются в низкоуровневых программных компонентах и модулях, чаще всего имеют ограниченное влияние. Симптомы, как правило, выявляются неподалеку от самой ошибки и/или одновременно с ней. Многие из этих ошибок могут быть обнаружены операционной системой или исполняемой резидентной частью компилятора, потому, что они вызывают переполнение памяти, обнаружимые ошибки указателя, чтение вне пределов файла или другие подобные дефекты.

Поток управления задает последовательность связей между узлами графа действий программы, он соответствует цепочке операторов программы, последовательно передающих друг другу управление вычислительным процессом. ***Построение графа потока управления программного компонента*** включает [Бе]***:***

- определение узлов:
  - каждому рассматриваемому объекту должен соответствовать один узел;
  - каждый узел должен иметь уникальное имя или идентификатор;
  - узлы могут обладать свойствами (весами), для каждого узла;
- определение связей:
  - каждая связь должна начинаться и заканчиваться на узле, предполагается наличие входного и/или выходного узла, если связь оборвана (то есть приходит ни откуда или ведет в никуда), то в построении модели допущена ошибка;
  - если между парой узлов существует больше одной связи, то целесообразно дать каждой связи свое уникальное имя;
  - каждой связи следует указать веса;
- в графе не обязательно присутствуют входной и выходной узлы, если они есть, то обозначьте все входные и выходные узлы соответственно;
- в графах не обязательно присутствуют циклы, но если они есть, то следует использовать специальные методы тестирования циклов, поэтому надо выделить все циклы.

Проверка узлов – необходима реализация достаточного количества тестов, для того чтобы убедиться, что все узлы именно такие, какими и должны быть по эталону, чтобы полностью выполнить проверку узлов следует применять:

- тесты, чтобы убедиться в наличии всех заданных узлов, ни один узел не должен быть пропущен;
- тесты, чтобы убедиться в отсутствии лишних узлов (это может потребовать неограниченного числа тестов и поэтому быть неосуществимо);
- тесты, чтобы убедиться в корректности весов узлов, если они есть;
- если существуют входной и выходной узлы, то в завершение предыдущих шагов можно выбрать пути от входа до выхода и выбрать входные значения так, чтобы программа проследовала по каждому из заданных в графе маршрутов.

Проверка связей, подразумевает, что выполнены все тесты проверки узлов и ***должна включать:***

- тесты, чтобы убедиться в наличии всех заданных связей – ни одна связь не пропущена;
- тесты, чтобы убедиться в отсутствии лишних связей (это может потребовать неограниченного числа тестов и поэтому быть неосуществимо);
- тестирование отношений: если отношение симметрично, то необходимо убедиться, что каждая связь является двусторонней; если отношение рефлексивно, то необходимо убедиться, что каждый узел связан сам с собой; если отношение транзитивно, то следует проверить его транзитивность в любой ситуации;
- если существуют входной и выходной узлы, то предыдущие шаги, как правило, завершаются подбором путей от входа к выходу, при этом выбираются такие входные параметры, с которыми программа ***проследует*** по выбранным маршрутам.

Если осуществляется проверка связей, то обычно осуществляется проверка узлов. Проверка связей в большинстве случаев требует большего количества тестов, чем проверка узлов. Если связи имеют веса, следует провести достаточное количество тестов, чтобы убедиться в правильности значения веса для каждой связи.

Поток данных тестирования компонентов задает последовательность узлов и связей преобразования переменных и констант при выполнении цепочек операторов программного компонента. При этом к элементам теории графов, представленным в выше, дополняются следующие понятия.

Входной узел графа потока данных, через который первично вводятся данные в программу, должен содержать имя входящего в этот узел объекта.

Константы следует рассматривать как вводы и как входные узлы. Они уязвимы для ошибок точно так же как и переменные и поэтому должны рассматриваться таким же образом.

Узел вывода – через который выводятся обработанные данные маршрута для последующего применения другим модулем.

Запоминающий узел – пара узлов для одного и того же объекта данных определяет значение переменной в памяти, сохраненное на диске, или значение в ОЗУ, предшествующее сохранению.

Обрабатывающий узел с одной или более входящими связями, которые обозначают объекты данных, и, по крайней мере, с одной исходящей связью. Исходящая связь обозначает вычисленную функцию этих объектов данных.

Значение связи в графах потока данных связаны с узлами, в которых эти значения вычисляются. Узел может быть использован для представления нескольких различных вычислений и, таким образом, может иметь несколько различных объектов, ассоциированных с исходящими связями.

Предикат выбора данных, – значение которого используется для выбора одного из нескольких объектов данных. Например, указатель на массив выбирает объект данных в массиве, на который указывает указатель.

Узел выбора данных, чьи входящие связи управляются предикатом выбора данных. Его значение выбирает объект данных, соответствующий входящей связи. Узел выбора данных вычисляет специальную функцию выбора значения входящей связи для использования в исходящей связи. Входящие связи узла выбора данных могут быть помечены условием предиката, при котором выбирается данная входящая связь.

Уровень файла – рассматривает весь файл как один объект данных, с которым обращаются как с единым целым, концентрируется на операциях, которые применяются к целому файлу и между файлами: открыть, закрыть, копировать, сравнить, переименовать, удалить, переместить, переслать, объединить, извлечь, создать.

Уровень записи – тесты, которые применяются в модели для уровня записи аналогичны операциям на уровне файла: добавить запись, удалить запись, копировать запись, найти запись, извлечь, сравнить, переместить.

Группы данных – в ряде случаев могут группироваться объекты данных, которые обрабатываются схожим образом, даже если они не связаны в контексте данного приложения. Одна и та же модель может быть использована для повторного использования в нескольких различных, возможно, не связанных, частях приложения.

Подграф – часть графа, которая соответствует правилам построения графов, то есть имеющая узлы входа и выхода, не имеет оборванных связей, не имеет изолированных узлов.

Порожденный подграф – выбранный в соответствии с определенным критерием так, что для данного критерия подграф обладает всеми свойствами полного графа для выбранных узлов и связей. Если, например, граф представляет собой граф потока управления и выбран критерий – поведение вдоль пути, то подграф содержит модель всего кода вдоль выбранного пути.

Порожденный подграф потока данных, общем случае, это подграф, содержащий все потоки данных, которые прямо или косвенно могут прийти в заданный узел, и все потоки данных, которые могут быть доступны из этого узла. Если порожденный подграф выбирается по отношению к выходному узлу, то в этом случае он содержит все узлы, которые могут повлиять на значение соответствующей выходной величины.

Основная литература

1. Бейзер Б. Тестирование черного ящика. Технология функционального тестирования программного обеспечения и систем. Пер. с англ. – СПб: ПИТЕР. 2004.
2. Блэк Р. Ключевые процессы тестирования. Пер. с англ. – М: ЛОРИ. 2006.
3. Канер С., Фолк Д., Нгуен Е. Тестирование программного обеспечения. Пер. с англ. – М: ДиаСофт. 2001.
4. Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. Пер. с англ. – М.: Вильямс. 2002.
5. Липаев В.В. Программная инженерия. Методологические основы. Учебник. – М.: ТЕИС. 2006.
6. Липаев В.В. Экономика производства сложных программных продуктов. – М.: СИНТЕГ. 2008.
7. Липаев В.В. Человеческие факторы в программной инженерии: рекомендации и требования к профессиональной квалификации специалистов. Учебник. – М.: СИНТЕГ. 2009.
8. Липаев В.В. Сертификация программных средств. Учебник. – М.: СИНТЕГ. 2010.
9. Соммервилл И. Инженерия программного обеспечения. 6-е издание. Пер. с англ. – М.: Вильямс. 2002.

Дополнительная литература

1. Вигерс К.И. Разработка требований к программному обеспечению. Пер. с англ. – М.: Русская редакция. 2004.
2. Гецци К., Джазайери М., Мандриоли Д. Основы инженерии программного обеспечения. Пер. с англ. – СПб.: БХВ-Петербург. 2005.
3. Дастин Э., Рэшка Д., Пол Д. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. Пер. с англ. – М. ЛОРИ. 2003.
4. Коберн А. Современные методы описания требований к системам. Пер. с англ. – М.: Лори.2002.
5. Тэллес М., Хсих Ю. Наука отладки. Пер. с англ. – М.: Кудиц-образ. 2003.

6. Уайт Б.А. Управление конфигурацией программных средств. Практическое руководство по Rational ClearCase. Пер. с англ. – М. ДМК Пресс. 2002.
7. Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И. Управление программными проектами: достижение оптимального качества при минимальных затратах. Пер. с англ. – М.: Вильямс. 2003.