Практическая работа №4

Тема: инфраструктура процесса тестирования

программных средств

Под инфраструктурой процесса тестирования понимаются:

* + выделение объектов тестирования;
  + проведение классификации ошибок для рассматриваемого класса тестируемых программ;
  + подготовка тестов, их выполнение и поиск разного рода ошибок и отказов в компонентах и в системе в целом;
  + служба проведения и управление процессом тестирования;
  + анализ результатов тестирования.

*Объекты тестирования* - компоненты, группы компонентов, подсистемы и система. Для каждого из них формируется стратегия проведения тестирования. Если объект тестирования относится к «белому ящику» или «черному ящику», состав компонентов которого неизвестный, то тестирование проводится посредством ввода в него входных тестовых данных для получения выходных данных. Стратегическая цель тестирования состоит в том, чтобы убедиться, что каждый рассматриваемый входной набор данных соответствует ожидаемым выходным данным. При таком подходе к тестированию не требуется знания внутренней структуры и логики объекта тестирования.

Проектировщик тестов должен заглянуть внутрь «черного ящика» и исследовать детали процессов обработки данных, вопросы обеспечения защиты и восстановления данных, а также интерфейс с другими программами и системами. Это способствует подготовке тестовых данных для проведения тестирования.

Для некоторых типов объектов группа тестирования не может сгенерировать представительное множество тестовых наборов, которые демонстрировали бы функциональную правильность работы компонента при всех возможных наборах тестов.

Поэтому предпочтительным является метод «белого ящика», при котором можно использовать структуру объекта для организации тестирования по различным ветвям. Например, можно выполнить тестовые наборы, которые проходят через все операторы или все контрольные точки компонента для того, чтобы убедиться в правильности их работы.

Методы поиска ошибок в программах

Международный стандарт ANSIЛEEE-729-83 разделяет все ошибки в разработке программ на следующие типы:

Ошибка (error) - состояние программы, при котором выдаются неправильные результаты, причиной которых являются изъяны (flaw) в операторах программы или в технологическом процессе ее разработки, что приводит к неправильной интерпретации исходной информации, следовательно, и к неверному решению.

Дефект (fault) в программе: следствие ошибок разработчика на любом из этапов разработки, которые могут содержаться в исходных или проектных спецификациях, текстах кодов программ, эксплуатационной документации и т.п. В процессе выполнения программы может быть обнаружен дефект или сбой.

Отказ (failure) - это отклонение программы от функционирования или невозможность программы выполнять функции, определенные требованиями и ограничениями, что рассматривается как событие, способствующее переходу программы в неработоспособное состояние из-за ошибок, скрытых в ней дефектов или сбоев в среде функционирования. Отказ может быть результатом следующих причин:

ошибочная спецификация или пропущенное требование, означающее, что спецификация точно не отражает того, что предполагал пользователь;

спецификация может содержать требование, которое невозможно выполнить на данной аппаратуре и программном обеспечении;

проект программы может содержать ошибки (например, база данных спроектирована без средств защиты от несанкционированного доступа пользователя, а требуется защита);

программа может быть неправильной, т.е. она выполняет несвойственный алгоритм или он реализован не полностью.

Таким образом, отказы, как правило, являются результатами одной или более ошибок в программе, а также наличия разного рода дефектов.

Ошибки на этапах процесса тестирования. Приведенные типы ошибок распределяются по этапам ЖЦ и им соответствуют следующие источники их возникновения:

* + - непреднамеренное отклонение разработчиков от рабочих стандартов или планов реализации;
    - спецификации функциональных и интерфейсных требований выполнены без соблюдения стандартов разработки, что приводит к нарушению функционирования программ;
    - организации процесса разработки - несовершенное или недостаточное управление руководителем проекта ресурсами (человеческими, техническими, программными и т.д.) и вопросами тестирования и интеграции элементов проекта.

1

Рассмотрим процесс тестирования, исходя из рекомендаций стандарта ISO/IEC 12207, и приведем типы ошибок, которые обнаруживаются на каждом процессе ЖЦ.

*Процесс разработки требований.* При определении исходной концепции системы иисходных требований к системе возникают ошибки аналитиков при спецификации верхнего уровня системы и построении концептуальной модели предметной области.

Характерными ошибками этого процесса являются:

* + - неадекватность спецификации требований конечным пользователям;
    - некорректность спецификации взаимодействия ПО со средой функционирования или с пользователями;
    - несоответствие требований заказчика к отдельным и общим свойствам ПО;
    - некорректность описания функциональных характеристик;
    - необеспеченность инструментальными средствами всех аспектов реализации требований заказчика и др.

*Процесс проектирования.* Ошибки при проектировании компонентов могут возникать при описании алгоритмов, логики управления, структур данных, интерфейсов, логики моделирования потоков данных, форматов ввода-вывода и др. В основе этих ошибок лежат дефекты спецификаций аналитиков и недоработки проектировщиков. К ним относятся ошибки, связанные:

* + - с определением интерфейса пользователя со средой;
    - описанием функций (неадекватность целей и задач компонентов, которые обнаруживаются при проверке комплекса компонентов);
    - определением процесса обработки информации и взаимодействия между процессами (результат некорректного определения взаимосвязей компонентов и процессов);
    - некорректным заданием данных и их структур при описании отдельных компонентов и ПС в целом;
    - некорректным описанием алгоритмов модулей;
    - определением условий возникновения возможных ошибок в программе;
    - нарушением принятых для проекта стандартов и технологий.

*Этап кодирования.* На данном этапе возникают ошибки, которые являются результатом дефектов проектирования, ошибок программистов и менеджеров в процессе разработки и отладки системы. Причиной ошибок являются:

* + - бесконтрольность значений входных параметров, индексов массивов, параметров циклов, выходных результатов, деления на 0 и др.;
    - неправильная обработка нерегулярных ситуаций при анализе кодов возврата от вызываемых подпрограмм, функций и др.;
    - нарушение стандартов кодирования (плохие комментарии, нерациональное выделение модулей и компонент и др.);
    - использование одного имени для обозначения разных объектов или разных имен одного объекта, плохая мнемоника имен;
    - несогласованное внесение изменений в программу разными разработчиками и др.

*Процесс тестирования.* На этом процессе ошибки допускаются программистами и тестировщиками при выполнении технологии сборки и тестирования, выбора тестовых наборов и сценариев тестирования и др. Отказы в программном обеспечении, вызванные такого рода ошибками, должны выявляться, устраняться и не отражаться на статистике ошибок компонентов и программного обеспечения в целом.

*Процесс сопровождения.* На процессе сопровождения обнаруживаются ошибки, причиной которых являются недоработки и дефекты эксплуатационной документации, недостаточные показатели модифицируемости и удобочитаемости, а также некомпетентность лиц, ответственных за сопровождение и/или усовершенствование ПО. В зависимости от сущности вносимых изменений на этом этапе могут возникать практически любые ошибки, аналогичные ранее перечисленным ошибкам на предыдущих этапах.

Все ошибки, которые возникают в программах, принято подразделять на следующие классы:

- логические и функциональные ошибки;

- ошибки вычислений и времени выполнения;

- ошибки ввода-вывода и манипулирования данными;

- ошибки интерфейсов;

- ошибки объема данных и др.

Логические ошибки являются причиной нарушения логики алгоритма, внутренней несогласованности переменных и операторов, а также правил программирования. Функциональные ошибки - следствие неправильно определенных функций, нарушения порядка их применения или отсутствия полноты их реализации и т.д.

Ошибки вычислений возникают по причине неточности исходных данных и реализованных формул, погрешностей методов, неправильного применения операций вычислений или операндов. Ошибки времени выполнения связаны с необеспечением требуемой скорости обработки запросов или времени восстановления программы.

Ошибки ввода-вывода и манипулирования данными являются следствием некачественной подготовки данных для выполнения программы, сбоев при занесении их в базы данных или при выборке из нее.

Ошибки интерфейса относятся к ошибкам взаимосвязи отдельных элементов друг с другом, что проявляется при передаче данных между ними, а также при взаимодействии со средой функционирования.

Ошибки объема относятся к данным и являются следствием того, что реализованные методы доступа и размеры баз данных не удовлетворяют реальным объемам информации системы или интенсивности их обработки.

Приведенные основные классы ошибок свойственны разным типам компонентов ПО, и проявляются они в программах по-разному. Так, при работе с БД возникают ошибки представления и манипулирования данными, логические ошибки в задании прикладных процедур обработки данных и др. В программах вычислительного характера преобладают ошибки вычислений, а в программах управления и обработки - логические и функциональные ошибки. В ПО, которое состоит из множества разноплановых программ, реализующих разные функции, могут содержаться ошибки разных типов. Ошибки интерфейсов и нарушение объема характерны для любого типа систем.

Анализ типов ошибок в программах является необходимым условием создания планов тестирования и методов тестирования для обеспечения правильности ПО.

На современном этапе развития средств поддержки разработки ПО (САSЕ-технологии, объектно-ориентированные методы и средства проектирования моделей и программ) проводится такое проектирование, при котором ПО защищается от наиболее типичных ошибок, и тем самым предотвращается появление программных дефектов.

**Связь ошибки с отказом.** Наличие ошибки в программе, как правило, приводит к отказу ПО при его функционировании. Для анализа причинно-следственных связей «ошибка-отказ» выполняются следующие действия:

- идентификация изъянов в технологиях проектирования и программирования;

- взаимосвязь изъянов процесса проектирования и допускаемых человеком ошибок;

- классификация отказов, изъянов и возможных ошибок, а также дефектов на каждом этапе разработки;

- сопоставление ошибок человека, допускаемых на определенном процессе разработки, и дефектов в объекте как следствий ошибок спецификации проекта, моделей программ;

- проверка и защита от ошибок на всех этапах ЖЦ, а также обнаружение дефектов на каждом этапе разработки;

- сопоставление дефектов и отказов в ПО для разработки системы взаимосвязей и методики локализации, сбора и анализа ин формации об отказах и дефектах;

- разработка подходов к процессам документирования и испытания ПО.

Конечная цель причинно-следственных связей «ошибка-отказ» заключается в определении методов и средств тестирования и обнаружения ошибок определенных классов, а также критериев завершения тестирования на множестве наборов данных; в определении путей совершенствования организации процесса разработки, тестирования и сопровождения ПО.

Приведем следующую классификацию типов отказов:

- аппаратный, при котором общесистемное ПО не работоспособно;

- информационный, вызванный ошибками во входных данных и передаче данных по каналам связи, а также при сбое устройств ввода (следствие аппаратных отказов);

- эргономический, вызванный ошибками оператора при его взаимодействии с машиной (этот отказ - вторичный отказ, может привести к информационному или функциональному отказам);

- программный, при наличии ошибок в компонентах и др.

Некоторые ошибки могут быть следствием недоработок при определении требований, проекта, генерации выходного кода или документации. С другой стороны, они порождаются в процессе разработки программы или при разработке интерфейсов отдельных элементов программы (нарушение порядка параметров, меньше или больше параметров и т.п.).

**Источники ошибок.** Ошибки могут быть порождены в процессе разработки проекта, компонентов, кода и документации. Как правило, они обнаруживаются при выполнении или сопровождении программного обеспечения в самых неожиданных и разных ее точках.

Некоторые ошибки в программе могут быть следствием недоработок при определении требований, проекта, генерации кода или документации. С другой стороны, ошибки порождаются в процессе разработки программы или интерфейсов ее элементов (например, при нарушении порядка задания параметров связи - меньше или больше, чем требуется и т.п.).

Причины появления ошибок - непонимание требований заказчика; неточная спецификация требований в документах проекта и др. Это приводит к тому, что реализуются некоторые функции системы, которые будут работать не так, как предлагает заказчик. В связи с этим проводится совместное обсуждение заказчиком и разработчиком некоторых деталей требований для их уточнения.

Команда разработчиков системы может также изменить синтаксис и семантику описания системы. Однако некоторые ошибки могут быть не обнаружены (например, неправильно заданы индексы или значения переменных этих операторов).

Классификация ошибок и тестов

Каждая организация, разрабатывающая ПО общесистемного назначения, сталкивается с проблемами нахождения ошибок. Поэтому приходится классифицировать типы обнаруживаемых ошибок и определять свое отношение к устранению этих ошибок.

На основе многолетней деятельности в области создания ПО разные фирмы создали свою классификацию ошибок, основанную на выявлении причин их появления в процессе разработки, в функциях и в области функциональной деятельности ПО.

Известно много различных подходов к классификации ошибок. Рассмотрим некоторые из них.

Фирма IВМ разработала подход к классификации ошибок, называемый ортогональной классификацией дефектов. Подход предусматривает разбиение ошибок по категориям с соответствующей ответственностью разработчиков за них.

Схема классификации не зависит от продукта, организации разработки и может применяться ко всем стадиям разработки ПО разного назначения. Таблица 1 дает список ошибок согласно данной классификации. Используя эту таблицу, разработчик имеет возможность идентифицировать не только типы ошибок, но и места, где пропущены или совершены ошибки. Предусмотрены ситуации, когда найдена неинициализированная переменная или инициализированной переменной присвоено неправильное значение.

*Таблица 1*

**Ортогональная классификация дефектов IBM**

|  |  |
| --- | --- |
| Контекст ошибки | Классификация дефектов |
| Функция | Ошибки интерфейсов конечных пользователей ПО, вызванные аппаратурой или связанные с глобальными структурами  данных |
| Интерфейс | Ошибки во взаимодействии с другими компонентами,  в вызовах, макросах, управляющих блоках или в списке параметров |
| Логика | Ошибки в программной логике, неохваченной валидацией, а также в использовании значений переменных |
| Присваивание | Ошибки в структуре данных или в инициализации переменных отдельных частей программы |
| Зацикливание | Ошибки, вызванные ресурсом времени, реальным временем или разделением времени |
|  |
| Среда | Ошибки в репозитории, в управлении изменениями или в контролируемых версиях проекта |
| Алгоритм | Ошибки, связанные с обеспечением эффективности, |
| корректности алгоритмов или структур данных системы | |
| Документация | Ошибки в записях документов сопровождения или в публикациях |

Ортогональность схемы классификации заключается в том, что любой ее термин принадлежит только одной категории.

Другими словами, прослеживаемая ошибка в системе должна находиться в одном из классов, что дает возможность разным разработчикам классифицировать ошибки одинаковым способом.

Фирма Hewlett-Packard использовала классификацию Буча, установив процентное соотношение ошибок, обнаруживаемых в ПО на разных стадиях разработки:

- в логике 32%;

- в вычислениях 18%;

- в коде 11%;

- в управлении данными 6%;

- в документации 19%;

- в требованиях 5%;

- в аппаратуре 4%;

- в интеграции 5%.

Это соотношение - типичное для многих фирм, производящих ПО, - имеет некоторые отклонения.

Исследования фирм IBM показали, чем позже обнаруживается ошибка в программе, тем дороже обходится ее исправление. Эта зависимость близка к экспоненциальной. Так, военно-воздушные силы США оценили стоимость разработки одной инструкции в 75 долл., а ее стоимость сопровождения составляет около 4000 долл.

Согласно данным стоимость анализа и формирования требований, внесения в них изменений составляет примерно 10 %, аналогично оценивается стоимость спецификации продукта. Стоимость кодирования оценивается более чем 20 %, а стоимость тестирования продукта составляет более 45 % от его общей стоимости. Значительную часть стоимости составляют сопровождение готового продукта и исправление обнаруженных в нем ошибок.

Определение теста. Для проверки правильности программ специально разрабатываются тесты и тестовые данные. Под тестом понимается некоторая программа, предназначенная для проверки работоспособности другой программы и обнаружения в ней ошибочных ситуаций. Тестовую проверку можно провести также путем введения в проверяемую программу отладочных операторов, которые будут сигнализировать о ходе ее выполнения и получения результатов.

Тестовые данные служат для проверки работы системы и составляются разными способами: генератором тестовых данных, проектной группой на основе вне машинных документов или имеющихся файлов, пользователем по спецификациям требований и др. Очень часто разрабатываются специальные формы входных документов, в которых отображается процесс выполнения программы с помощью тестовых данных.

Создаются тесты, проверяющие:

- полноту функций;

- согласованность интерфейсов;

- корректность выполнения функций и правильность функционирования системы в заданных условиях;

- надежность выполнения системы;

- защиту от сбоев аппаратуры и не выявленных ошибок и др.

Тестовые данные готовятся как для проверки отдельных программных элементов, так и для групп программ или комплексов на разных стадиях процесса разработки. На рис. 1. приведена классификация тестов проверки по объектам тестирования на основных этапах разработки.

Тестирование спецификаций

Тестирование программ

Тестирование комплекса

Тестирование при испытаниях

f!>

Тесты проверки:

-структуры программы;

- вычислений

•

и преобразования данных;

-полноты выполняемых функций

Тесты проверки:

-структуры комплекса;

* интерфейса

компонент;

* ограничений по памяти;
* длительности исполнения;
* полноты решения задач комплекса

Тесты проверки:

* соответствия требованиям;
* удобства установки рабочей версии;
* работы комплекса на оборудовании;
* удобства интерфейса пользователя;
* удобства модификации

и сопровождения

Тесты проверки:

* полноты

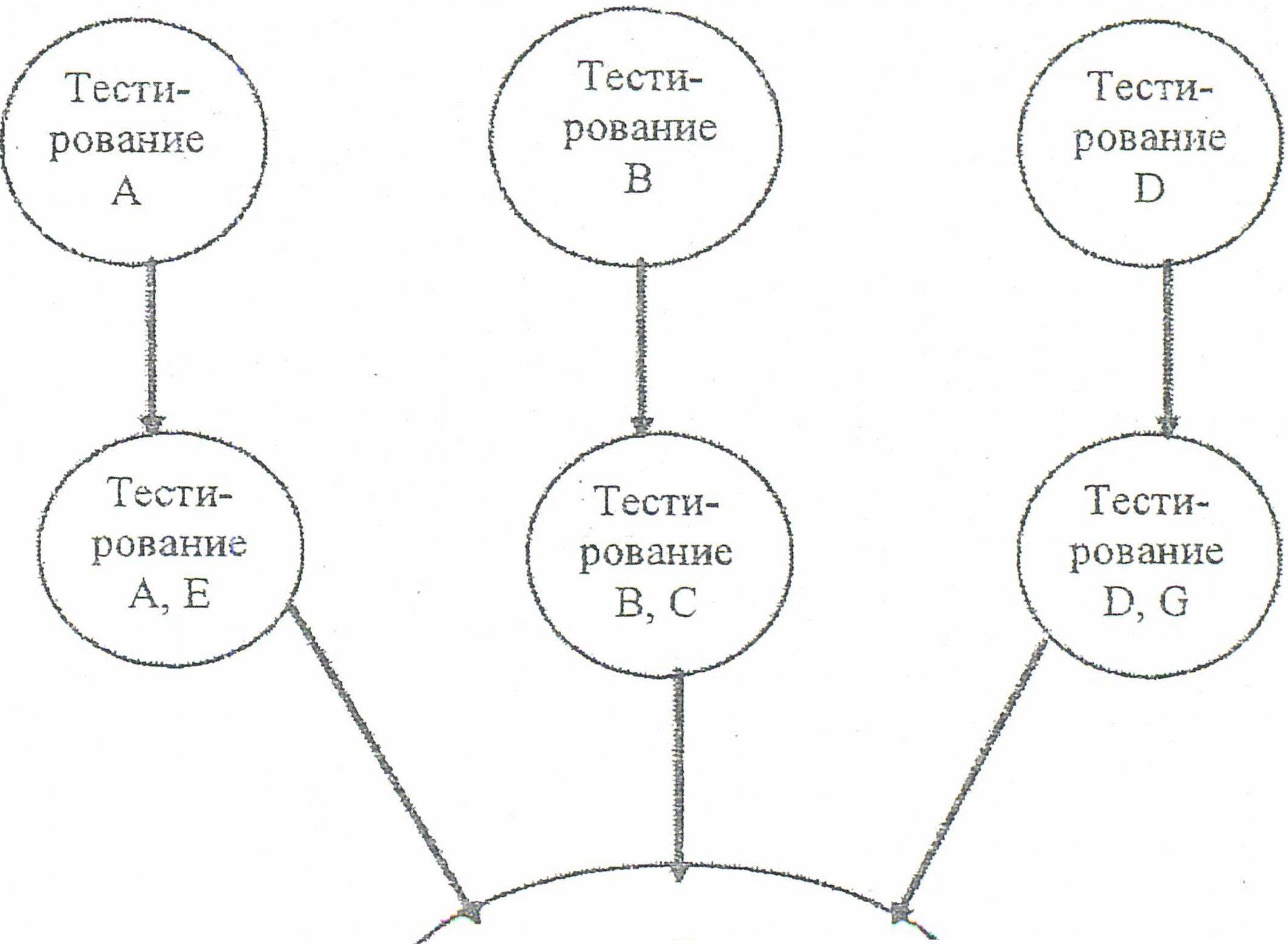
и согласованности функций;

* согласованности интерфейсов

Рис. 1. Классификация тестов проверок

Многие типы тестов готовятся заказчиком для проверки работы программной системы. Структура и содержание тестов зависят от вида тестируемого элемента, которым может быть модуль, компонент, группа компонентов, подсистема или система. Некоторые тесты зависят от цели и необходимости знать: работает ли система в соответствии с ее проектом, удовлетворены ли требования и участвует ли заказчик *в* проверке работы тестов и т.п.

В зависимости от задач, которые ставятся перед тестированием программ, составляются тесты проверки промежуточных результатов проектирования элементов на этапах ЖЦ, а также создаются тесты испытаний окончательного кода системы.

**Тесты интегрированной системы.** Тесты для проверки отдельных элементов системы и тесты интегрированной системы имеют общие и отличительные черты. Так, на рис. 2. в качестве примера приведена схема интеграции готовых оттестированных элементов. В ней заданы связи между разными уровнями тестирования интегрируемой ПС.

F

Тестировзние

\ А, В, С, D, Е, G /

Рис. 2. Интегрированное тестирование компонентов

/

Рассмотрим этот процесс более подробно. Каждый компонент этой схемы тестируется отдельно от других компонентов с помощью тестов, включающих наборы данных и сценарии, составленные в соответствии с их типами и функциями, специфицированные в проекте системы. Тестирование проводится в контрольной операционной среде на предопределенном множестве тестовых данных и операциях, производимых над ними.

Тесты обеспечивают проверку внутренней структуры, логики и граничных условий выполнения для каждого компонента.

Согласно приведенной схеме сначала тестируются компоненты А, В, D независимо друг от друга и каждый с отдельным тестом. После их проверки выполняется проверка интерфейсов для последую­ щей их интеграции, суть которой заключается в анализе выполнения операторов вызова А Е, В С, D G, на нижних уровнях графа: компоненты Е, С, G. При этом предполагается, что указанные вызываемые компоненты должны быть отлажены отдельно. Аналогично проверяются все обращения к компоненту F, являющемуся связывающим звеном с вышележащими элементами.

При этом могут возникать ошибки в случае неправильного задания параметров в операторах вызова или при вычислениях процедур или функций. Возникающие ошибки в связях устраняются, а за­ тем повторно проверяется связь с компонентом F в виде троек: компонент-интерфейс-компонент. Следующим шагом тестирования комплексной системы является проверка функционирования системы с помощью тестов проверки функций и требований к ним. После проверки системы на функциональных тестах происходит проверка на исполнительных и испытательных тестах, подготовленных согласно требованиям к ПО, аппаратуре и выполняемым функциям. Испытательному тесту предшествуют верификация и валидация ПО.

Тест испытаний системы в соответствии с требованиями заказчика проверяется в реальной среде, в которой система будет в дальнейшем функционировать.

Служба тестирования программных средств

За функциональные и исполнительные тесты несут ответственность разработчики, заказчик больше влияет на составление тестов испытаний и инсталляции системы.

Для этих целей, как правило, создается служба проверяющих ПС - команда тестировщиков, которая не зависит от штата разработчиков ПС. Некоторые члены этой команды - опытные профессионалы в этой области: аналитики, программисты, инженеры-тестировщики, которые посвящены в проблемы тестирования систем с начала разработки. Они имеют дело не только со спецификациями, но и с методами и средствами тестирования, организуют создание и выполнение тестов. С самого начала создания проекта тестировщики составляют планы тестирования, тестовые данные и сценарии, а также графики выполнения тестов.

Профессиональные тестировщики работают совместно с группой управления конфигурацией, чтобы обеспечить их документацией и другими механизмами для связи между собой тестов с требованиями проекта, конфигурацией и кодом. Они разрабатывают методы и процедуры тестирования. В эту команду включаются дополнительные специалисты, которые знакомы с требованиями системы или с подходами к их разработке. Аналитиков включают в команду, так как они понимают проблемы определения спецификаций заказчиков.

Многие специалисты сравнивают тестирование системы с созданием новой системы, в которой аналитики отражают потребности и цели заказчика, работая совместно с проектировщиками и добиваясь реализации идей и принципов работы системы. Проектировщики системы сообщают команде тестировщиков проектные цели, чтобы они знали декомпозицию системы на подсистемы и ее функции, а также принципы их работы. После проектирования тестов и тестовых покрытий команда тестировщиков проводит анализ возможностей системы.

Так как тесты и тестовые сценарии являются прямым отражением требований и проекта в целом, перспективы управления конфигурацией системы определяются именно этой командой. Обнаруживаемые в программе ошибки и изменения в системе отражаются в документации, требованиях, проекте, а также в описаниях входных и выходных данных или в других разрабатываемых артефактах. Вносимые изменения в процессе разработки приводят к модификации тестовых сценариев или в большей части к изменению планов тестирования. Специалисты по управлению конфигурацией учитывают эти изменения и координируют составление тестов.

В команду тестировщиков входят также пользователи. Они оценивают получаемые результаты, удобство использования, а также высказывают свое мнение о принципах работы системы.

Уполномоченные заказчика планируют работы до тех пор, пока используется и сопровождается система. При этом они могут привнести некоторые изменения в проект из-за неполноты заданных требований и сформулировать системные требования для проведения верификации системы и принятия решений о ее готовности и полезности.

План тестирования. Для проведения тестирования создается план (Test Plan), в котором описываются стратегии, ресурсы и график тестирования отдельных компонентов и системы в целом. В плане отмечаются работы для разных членов команды, которые выполняют определенные роли в этом процессе. План включает также определение роли тестов в каждом процессе, степень покрытия программы тестами и процент тестов, которые выполняются со специальными данными.

Тестовые инженеры создают множество тестовых сценариев (Test Cases), каждый из которых проверяет результат взаимодействия между актором и системой на основе пред- и постусловий использования таких сценариев. Сценарии в основном относятся к тестированию по типу «белого ящика» и ориентированы на проверку структуры и операций интеграции компонентов системы.

Для проведения тестирования тестовые инженеры предлагают процедуры тестирования (Test Procedures), включающие валидацию объектов и верификацию тестовых сценариев в соответствии с планом тестирования. Оценка тестов (Test Evaluation) заключается в оценке результатов тестирования, степени покрытия программ сценариями и статуса полученных ошибок.

Круг обязанностей инженера-тестировщика:

- программы тестов;

- тестовые ситуации;

- тестовые процедуры;

- оценка тестов;

- план теста.

Тестировщик интегрированной системы проводит тестирование интерфейсов и дает оценку результатов выполнения соответствующих тестов с помощью создаваемых им системных тестов, выполняет анализ результатов тестирования, проведенного с отдельными элементами системы. При выполнении системных тестов, как правило, находятся дефекты как результат глубоко скрытых по грешностей в программах, обнаруживаемых при длительной про гонке системы на тестовых данных и сценариях.

Управление процессом тестирования

Все способы тестирования ПС объединяются базой данных, где помещаются результаты тестирования системы. В ней содержатся все компоненты, тестовые контрольные данные, результаты тестирования и информация о документировании процесса тестирования.

База данных проекта поддерживается специальными инструментальными средствами типа CASE, которые обеспечивают ведение анализа ПО, сборку данных об их объектах, потоках данных и т.п. База данных проекта хранит также начальные и эталонные данные, которые используются для сопоставления данных, накопленных в базе, с данными, которые получены в процессе тестирования системы.

При тестировании выполняются разные виды расчетов характеристик этого процесса и способы планирования и управления:

1. Расчет продолжительности выполнения функций путем сбора средних показателей скорости выполнения операторов без выполнения программы на машине. Выявляются компоненты, которые требуют большого времени выполнения в реальной среде.

2. Управление выполнением тестирования путем подбора тестов проверки, их выполнения, селекции результатов тестирования и проведения сопоставления с эталонными значениями. Результаты данного процесса отображаются на дисплее, например, в графической форме (пути прохождения по графу программы), в виде диаграмм UML, данных об отказах и ошибках или конкретных значений исходных параметров программы. Эти данные анализируются разработчиками для формулирования выводов о направлениях дальнейшей проверки правильности программы или их завершении.

3. Планирование тестирования предназначено для распределения сроков работ по тестированию, распределения тестировщиков по отдельным видам работ и составления ими тестов проверки системы. Определяются стратегия и пути тестирования. В диалоге запрашиваются данные о реальных значениях процесса выполнения системы, структуре ветвления вершин графа и параметрах циклов. Проверенные циклы, как правило, изымаются из путей выполнения программы. При планировании путей выполнения создаются соответствующие тесты, критерии и входные значения.

Документирование результатов тестирования в соответствии с действующим стандартом ANSVIEEE 829 включает:

- описание задач, назначение и содержание ПС, а также перечень функций в соответствии с требованиями заказчика;

- технологии разработки системы;

- планы тестирования различных объектов, необходимых ресурсов, соответствующих специалистов для проведения тестирования и технологических способов;

- тесты, контрольные примеры, критерии и ограничения, оценки результатов программного продукта, а также процесса тестирования;

- учет процесса тестирования, составление отчетов об аномальных событиях, отказах и дефектах в итоговом документе системы.

**Задание:** ответить на контрольные вопросы.

## Контрольные вопросы

1. Продолжите: инфраструктура процесса тестирования - это ...
2. Перечислите типы ошибок по международному стандарту ANSVIEEE-729-83.
3. Назовите характерные ошибки процесса (этапа) разработки требований.
4. Перечислите ошибки процесса (этапа) проектирования.
5. Назовите классы ошибок, которые возникают впрограммах.
6. Какова классификация типов отказов ПО при его функционировании?
7. Какова классификация тестов проверки?
8. Что содержит план тестирования?
9. Обозначьте круг обязанностей инженера-тестировщика.