Лекция 1. Введение в отладку и тестирования программ

Основные характеристики качества и надежности программного продукта

1 Качество

Качество (quality) программного средства — это совокупность его черт и характеристик, которые влияют на его способность удовлетворять заданные потребности пользователей. Задача оценки качества программного средства опирается на необходимость формирования системы показателей, характеризующих качество функционирования программного средства, с учетом технологических возможностей разработчика. Для конкретных программных средств доминирующие критерии качества выделяются и определяются при проектировании требованиями технического задания и задачами функционирования программного средства.

2 Надежность

Надежность (reliability) программного средства — это его способность безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью. Альтернативой правильного ПП является надежное ПП. При этом под отказом в ПП понимают проявление в нем ошибки. Таким образом, надежное ПП не исключает наличия в нем ошибок - важно лишь, чтобы эти ошибки при практическом применении этого ПП в заданных условиях проявлялись достаточно редко. Убедиться, что ПП обладает таким свойством можно при его испытании путем тестирования, а также при практическом применении. Таким образом, фактически мы можем разрабатывать лишь надежные, а не правильные ПП. ГОСТ 27.002 – 89. Надёжность — свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных условиях применения, технического обслуживания, хранения и транспортирования.

Невозможно гарантировать отсутствие ошибок в программе. В лучшем случае можно попытаться показать наличие ошибок. Если программа правильно ведет себя для большого набора тестов, нет оснований утверждать, что в ней нет ошибок. Если считать, что набор тестов способен с большой вероятностью обнаружить возможные ошибки, то можно говорить о некотором уровне уверенности (надежности) в правильности работы программы, устанавливаемом этими тестами. Сформулируем следующее высказывание: если ваша цель - показать отсутствие ошибок, вы их найдете не слишком много. Если же ваша цель - показать наличие ошибок, вы найдете значительную их часть.

Термины и определения

Несмотря на кажущуюся схожесть, термины "тестирование", "верификация" и "валидация" означают разные уровни проверки корректности работы программной системы. Дабы избежать дальнейшей путаницы, четко определим эти понятия (Рис 1).

Тестирование (testing) программного обеспечения - вид деятельности в процессе разработки, который связан с выполнением процедур, направленных на обнаружение (доказательство наличия) ошибок (несоответствий, неполноты, двусмысленностей и т.д.) в текущем определении разрабатываемой программной системы. Процесс тестирования относится в первую очередь к проверке корректности программной реализации

системы, *соответствия реализации* требованиям, т.е. тестирование — это управляемое выполнение программы с целью обнаружения несоответствий ее поведения и требований.

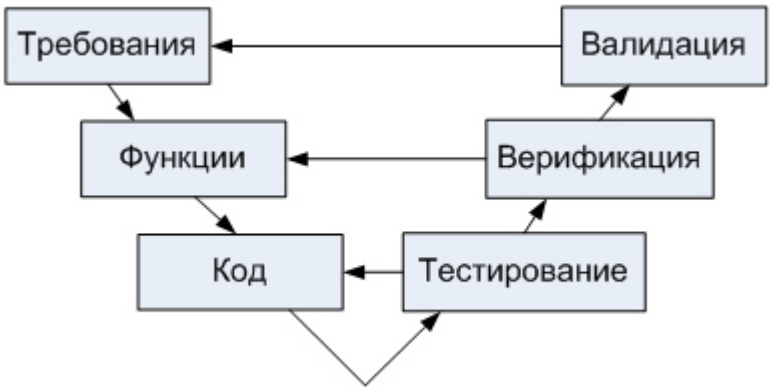

Рис. 1. Тестирование, верификация и валидация

Верификация (verification) программного обеспечения - более общее понятие, чем тестирование. Целью верификации является достижение гарантии того, что *верифицируемый* объект (требования или программный код) соответствует требованиям, реализован без непредусмотренных функций и удовлетворяет проектным спецификациям и стандартам. Процесс верификации включает в себя инспекции, тестирование кода, анализ результатов тестирования, формирование и анализ отчетов о проблемах. Таким образом, принято считать, что процесс тестирования является составной частью процесса верификации, такое же допущение сделано и в данном учебном курсе.

Испытание, валидация (validation) - попытка найти ошибки, выполняя программу в заданной реальной среде. программной системы - целью этого процесса является доказательство того, что в результате разработки системы мы достигли тех целей, которые планировали достичь благодаря ее использованию. Иными словами, валидация — это проверка соответствия системы ожиданиям заказчика. Вопросы, связанные с валидацией, выходят за рамки данного учебного курса и представляют собой отдельную интересную тему для изучения.

Если посмотреть на эти три процесса с точки зрения вопроса, на который они дают ответ, то тестирование отвечает на вопрос "Как это сделано?" или "Соответствует ли поведение разработанной программы требованиям?", верификация - "Что сделано?" или ""Соответствует ли разработанная система требованиям?", а валидация - "Сделано ли то, что нужно?" или "Соответствует ли разработанная система ожиданиям заказчика?".

Доказательство (proof) - попытка найти ошибки в программе безотносительно к внешней для программы среде. Большинство методов доказательства предполагает формулировку утверждений о поведении программы и доказательство математических теорем о правильности программы. Доказательства могут рассматриваться как форма тестирования, хотя они и не предполагают прямого выполнения программы.

Аттестация (certification) - авторитетное подтверждение правильности программы. При тестировании с целью аттестации выполняется сравнение с некоторым заранее определенным стандартом.

Отладка (debugging) не является разновидностью тестирования. Хотя слова «отладка» и «тестирование» часто используются как синонимы, но под ними подразумеваются разные виды деятельности. Тестирование — это деятельность, направленная на обнаружение ошибок. Отладка направлена на установление точной природы известной ошибки, а затем на исправление этой ошибки. Эти два вида деятельности связаны, т.к. результаты тестирования являются исходными данными для отладки.

Тестирование модуля, или автономное тестирование (module testing, unit testing) - контроль отдельного программного модуля, обычно в изолированной среде (изолированно от всех остальных модулей).

Тестирование сопряжений (integration testing) - контроль сопряжений между частями системы (модулями, компонентами, подсистемами).

Тестирование внешних функций (external function testing) - контроль внешнего поведения, определенного внешними спецификациями.

Комплексное тестирование (system testing) - контроль и/или испытание системы по отношению к исходным целям.

Комплексное тестирование является процессом контроля, если оно выполняется в моделируемой среде, и процессом испытания, если выполняется в реальной среде.

Тестирование приемлемости (acceptance testing) - проверка соответствия программы требованиям пользователя.

Тестирование настройки (installation testing) - проверка соответствия каждого конкретного варианта установки системы с целью выявить любые ошибки, возникшие в процессе настройки системы.

Стресс - тестирование (Stress Testing) — один из видов тестирования программного обеспечения, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования. Стресс-тестирование особенно необходимо для «критически важного» ПО, однако также используется и для остального ПО. Обычно стресс-тестирование лучше обнаруживает устойчивость, доступность и обработку исключений системой под большой нагрузкой, чем то, что считается корректным поведением в нормальных условиях.

Отладка программного средства. Основные понятия

Отладка программного средства (ПП) — это деятельность, направленная на обнаружение и исправление ошибок в ПП с использованием процессов выполнения его программ. Отладку можно представить в виде многократного повторения трех процессов:

тестирования, в результате которого может быть констатировано наличие в ПП ошибки, поиска места ошибки в программах и документации ПП и редактирования программ и документации с целью устранения обнаруженной ошибки. Другими словами:

Отладка = Тестирование + Поиск ошибок + Редактирование.

Процесс отладки включает:

- действия, направленные на выявление ошибок (тестирование);
- диагностику и локализацию ошибок (определение характера ошибок и их местонахождение);
- внесение исправлений в программу с целью устранения ошибок.

Из трех перечисленных видов работ самым трудоемким и дорогим является тестирование, затраты на которое приближаются к 45 % общих затрат на разработку ПП.

Надежность невозможно внести в программу в результате тестирования, она определяется правильностью этапов проектирования. Наилучшее решение проблемы надежности - с самого начала не допускать ошибок в программе. Однако вероятность того, что удастся безупречно спроектировать большую программу, мала. Роль тестирования состоит в том, чтобы определить местонахождение немногочисленных ошибок, оставшихся в хорошо спроектированной программе. Попытки с помощью тестирования достичь надежности плохо спроектированной программы безнадежны.

.

Принципы и виды отладки программного средства

Успех отладки ПП в значительной степени предопределяет рациональная организация тестирования. При отладке ПП отыскиваются и устраняются, в основном, те ошибки, наличие которых в ПП устанавливается при тестировании. Тестирование не может доказать правильность ПП, в лучшем случае оно может продемонстрировать наличие в нем ошибки. Другими словами, нельзя гарантировать, что тестированием ПП практически выполнимым набором тестов можно установить наличие каждой имеющейся в ПП ошибки. Поэтому возникает две задачи. Первая задача: подготовить такой набор тестов и применить к ним ПП, чтобы обнаружить в нем по возможности большее число ошибок. Однако чем дольше продолжается процесс тестирования (и отладки в целом), тем большей становится стоимость ПП. Вторая задача: определить момент окончания отладки ПП (или отдельной его компоненты). Признаком возможности окончания отладки является полнота охвата, пропущенными через ПП, тестами множества различных ситуаций, возникающих при выполнении программ ПП, и относительно редкое проявление ошибок в ПП на последнем отрезке процесса тестирования. Последнее определяется в соответствии с требуемой степенью надежности ПП, указанной в спецификации его качества.

Для оптимизации набора тестов, т.е. для подготовки такого набора тестов, который позволял бы при заданном их числе (или при заданном интервале времени, отведенном на тестирование) обнаруживать большее число ошибок в ПП, необходимо, во-первых, заранее планировать этот набор и, во-вторых, использовать рациональную стратегию планирования (проектирования) тестов. Проектирование тестов можно начинать сразу же после

завершения этапа внешнего описания ПП. Возможны разные подходы к выработке стратегии проектирования тестов, которые можно условно графически разместить (рис. 17) между следующими двумя крайними подходами.

Рис. 17. Подходы к проектированию тестов

Левый крайний подход заключается в том, что тесты проектируются только на основании изучения спецификаций ПП (внешнего описания, описания архитектуры и спецификации модулей). Строение модулей при этом никак не учитывается, т.е. они рассматриваются как черные ящики. Фактически такой подход требует полного перебора всех наборов входных данных, так как в противном случае некоторые участки программ ПП могут не работать при пропуске любого теста, а это значит, что содержащиеся в них ошибки не будут проявляться. Однако тестирование ПП полным множеством наборов входных данных практически неосуществимо. Правый крайний подход заключается в том, что тесты проектируются на основании изучения текстов программ с целью протестировать все пути выполнения каждой программ ПП. Если принять во внимание наличие в программах циклов с переменным числом повторений, то различных путей выполнения программ ПП может оказаться много, так что их тестирование будет практически неосуществимо.

Оптимальная стратегия проектирования тестов расположена внутри интервала между этими крайними подходами, но ближе к левому краю. Она включает проектирование значительной части тестов по спецификациям, но требует также проектирования некоторых тестов и по текстам программ. При этом в первом случае стратегия базируется на принципах:

- на каждую используемую функцию или возможность - хотя бы один тест;

- на каждую область и на каждую границу изменения какой-либо входной величины - хотя бы один тест;

- на каждую особую (исключительную) ситуацию, указанную в спецификациях - хотя бы один тест.

Во втором случае стратегия базируется на принципе: каждая команда каждой программы ПП должна проработать хотя бы на одном тесте.

Оптимальную стратегию проектирования тестов можно конкретизировать на основании следующего принципа: для каждого программного документа (включая тексты программ), входящего в состав ПП, должны проектироваться свои тесты с целью выявления в нем ошибок. Во всяком случае, этот принцип необходимо соблюдать в соответствии с определением ПП и содержанием понятия технологии программирования как технологии разработки надежных ПП. Различают два основных вида отладки (включая тестирование): автономную и комплексную отладку ПП.

Автономная отладка ПП означает последовательное раздельное тестирование различных частей программ, входящих в ПП, с поиском и исправлением в них фиксируемых при тестировании ошибок. Она фактически включает отладку каждого программного модуля и отладку сопряжения модулей.

Комплексная отладка означает тестирование ПП в целом с поиском и исправлением фиксируемых при тестировании ошибок во всех документах (включая тексты программ ПП), относящихся к ПП в целом. К таким документам относятся определение требований к ПП, спецификация качества ПП, функциональная спецификация ПП, описание архитектуры ПП и тексты программ ПП.

Принципы отладки программного продукта

Приводятся общие рекомендации по организации отладки ПП. Но сначала следует отметить некоторый феномен, который подтверждает важность предупреждения ошибок на предыдущих этапах разработки: по мере роста числа обнаруженных и исправленных ошибок в ПП растет также относительная вероятность существования в нем необнаруженных ошибок. Это объясняется тем, что при росте числа ошибок, обнаруженных в ПП, уточняется и наше представление об общем числе допущенных в нем ошибок, а значит, в какой-то мере, и о числе необнаруженных еще ошибок.

Принцип 1. Считайте тестирование ключевой задачей разработки ПП, поручайте его самым квалифицированным и одаренным программистам; нежелательно тестировать свою собственную программу.

Принцип 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Принцип 3. Готовьте тесты, как для правильных, так и для неправильных данных.

Принцип 4. Документируйте пропуск тестов через компьютер; детально изучайте результаты каждого теста; избегайте тестов, пропуск которых нельзя повторить.

Принцип 5. Каждый модуль подключайте к программе только один раз; никогда не изменяйте программу, чтобы облегчить ее тестирование.

Принцип 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПП или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

Существуют несколько эмпирических правил проведения отладки и тестирования программ, обобщающих опыт тестировщиков.

1. Процесс тестирования более эффективен, если проводится не автором программы. По своей сути тестирование — это процесс деструктивный (разрушительный). Именно этим и объясняется, почему многие считают его трудным. Особенно трудным и малоэффективным он является для самого автора программы, так как после выполнения конструктивной части при проектировании и написания программы, ему трудно перестроиться на деструктивный образ мышления и, создав программу, тут же приступить к пристрастному выявлению в ней ошибок. Поэтому для проведения тестирования

создаются специальные группы тестирования. Это не означает, что программист не может тестировать свою программу. Речь идет о повышении эффективности тестирования.

2. Необходимой частью тестового набора данных должно быть описание предполагаемых значений результатов тестовых прогонов. Тестирование как процесс многократного выполнения программы проводится на многочисленных входных наборах данных. Чтобы определить правильность полученных в результате очередного тестового прогона данных, необходимо знать ожидаемый результат. Таким образом, тестовый набор данных должен включать в себя два компонента: описание входных данных, описание точного и корректного результата, соответствующего набору входных данных. Этот принцип сложно, а в некоторых случаях и невозможно реализовать на практике. Сложность его заключается в том, что при тестировании программы (модуля) необходимо для каждого входного набора данных рассчитать вручную ожидаемый результат или найти допустимый интервал изменения выходных данных. Процесс этот трудоемкий даже для небольших программ, так как он требует ручных расчетов, следуя логике алгоритма программы. Из рассмотренного принципа, который трудно реализуем, но которого следует придерживаться логически, вытекает следующий.

3. Необходимо изучить результаты каждого теста. Из практики следует, что значительная часть обнаруженных ошибок могла быть выявлена в результате первых тестовых прогонов, но они были пропущены вследствие недостаточно тщательного анализа их результатов.

4. Тесты для неправильных и непредусмотренных входных данных должны разрабатываться также тщательно, как для правильных и предусмотренных. Согласно этому принципу при обработке данных, выходящих за область допустимых значений, в тестируемой программе должна быть предусмотрена диагностика в виде сообщений. Если сообщение о причине невозможности обработки по предложенному алгоритму отсутствует, и программа завершается аварийно или ведет себя непредсказуемо, то такая программа не может считаться работоспособной и требует существенной доработки. Тестовые наборы данных из области недопустимых входных значений обладают большей обнаруживающей способностью, чем тесты, соответствующие корректным входным данным.

5. Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и не делает ли она того, чего не должна делать. Это утверждение логически вытекает из предыдущего. Необходимо любую программу проверить на нежелательные побочные эффекты.

6. Следует тщательнее проверять те участки программ, где обнаруживается больше ошибок. Утверждается, что вероятность наличия необнаруженных ошибок в какой-либо части программы пропорциональна числу ошибок, уже обнаруженных в этой части. Возможно, что те части программы, где при тестировании обнаружено большее число ошибок, либо были слабо проработаны с точки зрения системного анализа, либо разрабатывались программистами более низкой квалификации.

Автономная отладка программного средства

При автономной отладке ПП каждый модуль на самом деле тестируется в некотором программном окружении, кроме случая, когда отлаживаемая программа состоит только из одного модуля. Это окружение состоит из других модулей, часть которых является

модулями отлаживаемой программы, которые уже отлажены, а часть - модулями, управляющими отладкой (отладочными модулями). Таким образом, при автономной отладке всегда тестируется некоторая программа (тестируемая программа), построенная специально для тестирования отлаживаемого модуля. Эта программа лишь частично совпадает с отлаживаемой программой, кроме случая, когда отлаживается последний модуль отлаживаемой программы. В процессе автономной отладки ПП производится наращивание тестируемой программы отлаженными модулями: при переходе к отладке следующего модуля в его программное окружение добавляется последний отлаженный модуль. Такой процесс наращивания программного окружения отлаженными модулями называется интеграцией программы. Отладочные модули, входящие в окружение отлаживаемого модуля, зависят от порядка, в каком отлаживаются модули этой программы, от того, какой модуль отлаживается и, возможно, от того, какой тест будет пропускаться.

При восходящем тестировании это окружение будет содержать только один отладочный модуль (кроме случая, когда отлаживается последний модуль отлаживаемой программы), который будет головным в тестируемой программе. Такой отладочный модуль называют ведущим (или драйвером). Ведущий отладочный модуль подготавливает информационную среду для тестирования отлаживаемого модуля (т.е. формирует ее состояние, требуемое для тестирования этого модуля, в частности путем ввода некоторых тестовых данных), осуществляет обращение к отлаживаемому модулю и после окончания его работы выдает необходимые сообщения. При отладке одного модуля для разных тестов могут составляться разные ведущие отладочные модули.

При нисходящем тестировании окружение отлаживаемого модуля в качестве отладочных модулей содержит отладочные имитаторы (заглушки) некоторых еще не отлаженных модулей. К таким модулям относятся, прежде всего, все модули, к которым может обращаться отлаживаемый модуль, а также еще не отлаженные модули, к которым могут обращаться уже отлаженные модули (включенные в это окружение). Некоторые из этих имитаторов при отладке одного модуля могут изменяться для разных тестов.

На практике в окружении отлаживаемого модуля могут содержаться отладочные модули обоих типов, если используется смешанная стратегия тестирования. Это связано с тем, что как восходящее, так и нисходящее тестирование имеет свои достоинства и свои недостатки.

К достоинствам восходящего тестирования относятся:

- простота подготовки тестов;

- возможность полной реализации плана тестирования модуля.

Это связано с тем, что тестовое состояние информационной среды готовится непосредственно перед обращением к отлаживаемому модулю (ведущим отладочным модулем).

Недостатками восходящего тестирования являются следующие его особенности:

- тестовые данные готовятся, как правило, не в той форме, которая рассчитана на пользователя (кроме случая, когда отлаживается последний, головной, модуль отлаживаемой программы);

• большой объем отладочного программирования (при отладке одного модуля приходится составлять много ведущих отладочных модулей, формирующих подходящее состояние информационной среды для разных тестов);

• необходимость специального тестирования сопряжения модулей.

К достоинствам нисходящего тестирования относятся следующие его особенности:

• большинство тестов готовится в форме, рассчитанной на пользователя;

• во многих случаях относительно небольшой объем отладочного программирования (имитаторы модулей, как правило, весьма просты и каждый пригоден для большого числа, нередко - для всех, тестов);

• отпадает необходимость тестирования сопряжения модулей.

Недостатком нисходящего тестирования является то, что тестовое состояние информационной среды перед обращением к отлаживаемому модулю готовится косвенно, оно является результатом применения уже отлаженных модулей к тестовым данным или данным, выдаваемым имитаторами. Это, во-первых, затрудняет подготовку тестов и требует высокой квалификации тестовика (разработчика тестов), а во-вторых, делает затруднительным или даже невозможным реализацию полного плана тестирования отлаживаемого модуля. Указанный недостаток иногда вынуждает разработчиков применять восходящее тестирование даже в случае нисходящей разработки. Однако чаще применяют некоторые модификации нисходящего тестирования, либо некоторую комбинацию нисходящего и восходящего тестирования. Исходя из того, что нисходящее тестирование, в принципе, является предпочтительным, остановимся на приемах, позволяющих в какой-то мере преодолеть указанные трудности.

Прежде всего, необходимо организовать отладку программы таким образом, чтобы как можно раньше были отлажены модули, осуществляющие ввод данных, тогда тестовые данные можно готовить в форме, рассчитанной на пользователя, что существенно упростит подготовку последующих тестов. Далеко не всегда этот ввод осуществляется в головном модуле, поэтому приходится в первую очередь отлаживать цепочки модулей, ведущие к модулям, осуществляющим указанный ввод. Пока модули, осуществляющие ввод данных, не отлажены, тестовые данные поставляются некоторыми имитаторами: они либо включаются в имитатор как его часть, либо вводятся этим имитатором.

При нисходящем тестировании некоторые состояния информационной среды, при которых требуется тестировать отлаживаемый модуль, могут не возникать при выполнении отлаживаемой программы ни при каких входных данных. В этих случаях можно было бы вообще не тестировать отлаживаемый модуль, так как обнаруживаемые при этом ошибки не будут проявляться при выполнении отлаживаемой программы ни при каких входных данных. Однако так поступать не рекомендуется, так как при изменениях отлаживаемой программы (например, при сопровождении ПП) не использованные для тестирования отлаживаемого модуля состояния информационной среды могут уже возникать, что требует дополнительного тестирования этого модуля (а этого при рациональной организации отладки можно было бы не делать, если сам данный модуль не изменялся). Для осуществления тестирования отлаживаемого модуля в указанных ситуациях иногда используют подходящие имитаторы, чтобы создать требуемое состояние информационной среды. Чаще же пользуются модифицированным вариантом нисходящего тестирования,

при котором отлаживаемые модули перед их интеграцией предварительно тестируются отдельно (в этом случае в окружении отлаживаемого модуля появляется ведущий отладочный модуль, наряду с имитаторами модулей, к которым может обращаться отлаживаемый модуль). Однако представляется более целесообразной другая модификация нисходящего тестирования: после завершения нисходящего тестирования отлаживаемого модуля для достижимых тестовых состояний информационной среды следует его отдельно протестировать для остальных требуемых состояний информационной среды.

Часто применяют также комбинацию восходящего и нисходящего тестирования, которую называют методом сандвича. Сущность этого метода заключается в одновременном осуществлении как восходящего, так и нисходящего тестирования, пока эти два процесса тестирования не встретятся на каком-либо модуле где-то в середине структуры отлаживаемой программы. Этот метод при разумном порядке тестирования позволяет воспользоваться достоинствами как восходящего, так и нисходящего тестирования, а также в значительной степени нейтрализовать их недостатки.

Весьма важным при автономной отладке является тестирование сопряжения модулей. Дело в том, что спецификация каждого модуля программы, кроме головного, используется в этой программы в двух ситуациях: во-первых, при разработке текста этого модуля и, во-вторых, при написании обращения к этому модулю в других модулях программы. И в том, и в другом случае в результате ошибки может быть нарушено требуемое соответствие заданной спецификации модуля. Такие ошибки требуется обнаруживать и устранять. Для этого и предназначено тестирование сопряжения модулей. При нисходящем тестировании тестирование сопряжения осуществляется попутно каждым пропускаемым тестом, что считают достоинством нисходящего тестирования. При восходящем тестировании обращение к отлаживаемому модулю производится не из модулей отлаживаемой программы, а из ведущего отладочного модуля. В связи с этим существует опасность, что последний модуль может приспособиться к некоторым «заблуждениям» отлаживаемого модуля. Поэтому, приступая (в процессе интеграции программы) к отладке нового модуля, приходится тестировать каждое обращение к ранее отлаженному модулю с целью обнаружения несогласованности этого обращения с телом соответствующего модуля (и не исключено, что виноват в этом ранее отлаженный модуль). Таким образом, приходится частично повторять в новых условиях тестирование ранее отлаженного модуля, при этом возникают те же трудности, что и при нисходящем тестировании.

Автономное тестирование модуля целесообразно осуществлять в четыре последовательно выполняемых шага.

1. На основании спецификации отлаживаемого модуля подготовьте тесты для каждой возможности и каждой ситуации, для каждой границы областей допустимых значений всех входных данных, для каждой области изменения данных, для каждой области недопустимых значений всех входных данных и каждого недопустимого условия.

2. Проверьте текст модуля, чтобы убедиться, что каждое направление любого разветвления будет пройдено хотя бы на одном тесте. Добавьте недостающие тесты.

3. Проверьте текст модуля, чтобы убедиться, что для каждого цикла существуют тесты, обеспечивающие, по крайней мере, три следующие ситуации: тело цикла не выполняется ни разу, тело цикла выполняется один раз и тело цикла выполняется максимальное число раз. Добавьте недостающие тесты.

4. Проверьте текст модуля, чтобы убедиться, что существуют тесты, проверяющие чувствительность к отдельным особым значениям входных данных. Добавьте недостающие тесты.

Комплексная отладка программного средства. При комплексной отладке тестируется ПП в целом, причем тесты готовятся по каждому из документов ПП. Тестирование этих документов производится, как правило, в порядке, обратном их разработке. Исключение составляет лишь тестирование документации по применению, которая разрабатывается по внешнему описанию параллельно с разработкой текстов программ — это тестирование лучше производить после завершения тестирования внешнего описания. Тестирование при комплексной отладке представляет собой применение ПП к конкретным данным, которые могут возникнуть у пользователя (в частности, все тесты готовятся в форме, рассчитанной на пользователя), но, возможно, в моделируемой (а не в реальной) среде. Например, некоторые недоступные при комплексной отладке устройства ввода и вывода могут быть заменены их программными имитаторами.

Тестирование архитектуры ПП. Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ ПП. К моменту начала тестирования архитектуры ПП должна быть уже закончена автономная отладка каждой подсистемы. Ошибки реализации архитектуры могут быть связаны, прежде всего, с взаимодействием этих подсистем, в частности, с реализацией архитектурных функций (если они есть). Поэтому хотелось бы проверить все пути взаимодействия между подсистемами ПП. При этом желательно хотя бы протестировать все цепочки выполнения подсистем без повторного вхождения последних. Если заданная архитектура представляет ПП в качестве малой системы из выделенных подсистем, то число таких цепочек будет вполне обозримо.

Тестирование внешних функций. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ ПП. Несмотря на то, что все эти программы автономно уже отлажены, указанные расхождения могут быть, например, из-за несоответствия внутренних спецификаций программ и их модулей (на основании которых производилось автономное тестирование) функциональной спецификации ПП. Как правило, тестирование внешних функций производится так же, как и тестирование модулей на первом шаге, т.е. как черного ящика.

Тестирование качества ПП. Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества ПП. Это наиболее трудный и наименее изученный вид тестирования. Далеко не каждый примитив качества ПП может быть испытан тестированием. Завершенность ПП проверяется уже при тестировании внешних функций. На данном этапе тестирование этого примитива качества может быть продолжено, если требуется получить какую-либо вероятностную оценку степени надежности ПП. Однако методика такого тестирования еще требует своей разработки. Могут тестироваться такие примитивы качества, как точность, устойчивость, защищенность, временная эффективность, в какой-то мере эффективность по памяти, эффективность по устройствам, расширяемость и, частично, независимость от устройств. Каждый из этих видов тестирования имеет свою специфику и заслуживает отдельного рассмотрения. Легкость применения ПП (критерий качества, включающий несколько примитивов качества) оценивается при тестировании документации по применению ПП.

Тестирование документации по применению ПП. Целью тестирования является поиск несогласованности документации по применению и совокупностью программ ПП,

а также выявление неудобств, возникающих при применении ПП. Этот этап непосредственно предшествует подключению пользователя к завершению разработки ПП (тестированию определения требований к ПП и аттестации ПП), поэтому разработчикам важно, сначала самим воспользоваться ПП так, как это будет делать пользователь. Все тесты на этом этапе готовятся на основании только документации по применению ПП. Прежде всего, должны тестироваться возможности ПП, как это делалось при тестировании внешних функций, но только на основании документации по применению. Должны быть протестированы все неясные места в документации, а также все примеры, использованные в документации. Далее тестируются наиболее трудные случаи применения ПП с целью обнаружить нарушение требований относительности легкости применения ПП.

Тестирование определения требований к ПП. Целью тестирования является выяснение, в какой мере ПП не соответствует предъявленному определению требований к нему. Особенность этого вида тестирования заключается в том, что его осуществляет организация-покупатель или организация-пользователь ПП как один из путей преодоления барьера между разработчиком и пользователем. Обычно это тестирование производится с помощью контрольных (типовых) задач, для которых известен результат решения. В тех случаях, когда разрабатываемое ПП должно прийти на смену другой версии ПП, которая решает хотя бы часть задач разрабатываемого ПП, тестирование производится путем решения общих задач с помощью, как старого, так и нового ПП (с последующим сопоставлением полученных результатов). Иногда в качестве формы такого тестирования используют опытную эксплуатацию ПП - ограниченное применение нового ПП с анализом использования результатов в практической деятельности. По существу, этот вид тестирования во многом перекликается с испытанием ПП при его аттестации, но выполняется до аттестации, а иногда и вместо аттестации.

Испытание программных продуктов.

Под испытанием программной продукции следует понимать экспериментальное определение количественных и/или качественных характеристик свойств продукции при ее функционировании в реальной среде и/или моделировании среды функционирования.

Целью испытания является экспериментальное определение фактических характеристик свойств испытываемого программного изделия (ПИ). Эти характеристики могут быть как количественными, так и качественными. Важно, чтобы на их основе можно было сделать вывод о пригодности ПИ к использованию по своему назначению. Если вывод отрицательный, то образец ПИ возвращается на доработку.

Испытание является завершающим этапом разработки. Ему предшествует этап статической и динамической отладки программ. Основным методом динамической отладки является тестирование. В узком смысле цель тестирования состоит в обнаружении ошибок, цель же отладки - не только в обнаружении, но и в устранении ошибок. Однако ограничиться только отладкой программы, если есть уверенность в том, что все ошибки в ней устранены, нельзя. Цели у отладки и испытания разные. Полностью отлаженная программа может не обладать определенными потребительскими свойствами и тем самым быть непригодной к использованию. Не может служить альтернативой испытанию и проверка работоспособности программы на контрольном примере, так как программа, работоспособная в условиях контрольного примера, может оказаться неработоспособной в других условиях применения.

В соответствии с ГОСТ 19.004-80 под испытанием программ понимают установление соответствия программы заданным требованиям и программным

документам. Это определение построено на предположении, что в техническом задании на разработку программы определены все требования (характеристики), обеспечение которых гарантирует пригодность программы к использованию по своему назначению.

При отсутствии технического задания (ТЗ) на разработку программного средства (ПП) или полного и обоснованного перечня требований к характеристикам, разрабатываемого ПП задача испытания ПП становится неопределенной и неконструктивной.

Длительность испытания зависит от типа, конфигурации (сложности) программного средства, а также от целей и степени автоматизации рассматриваемого технологического процесса (например, при испытании операционных систем от одного до шести месяцев). Сложные программные комплексы после интеграции могут испытываться и более длительное время.

Основными видами испытания программных продуктов (ПП) являются:

- предварительные;
- приемочные;
- эксплуатационные испытания, включая опытную эксплуатацию.

В зависимости от места проведения различают стендовые и полигонные испытания.

Под испытательным стендом понимают совокупность технических устройств и математических моделей, обеспечивающих в автоматическом режиме имитацию среды функционирования; поступление входных данных, искажающие воздействия; регистрацию информации о функционировании ПП, а также управление процессом испытания и объектом испытания.

Если в основу стендовых испытаний положен принцип моделирования, то соответствующие испытательные стенды называют моделирующими.

Испытательным полигоном называют место, предназначенное для испытаний в условиях, близких к условиям эксплуатации, и обеспеченное необходимыми средствами испытания.

Полигонным испытаниям подвергают системы, работающие в реальном масштабе времени. В полигонных условиях обычно сочетают натурные испытания с использованием реальных объектов автоматизируемых систем и моделирование некоторых объектов и процессов их функционирования.

По степени зависимости испытателей от разработчиков различают зависимые и независимые испытания.

При зависимых испытаниях основные операции с испытываемыми ПП (подготовка к работе, подготовка и ввод исходных данных, регистрация и анализ результатов) выполняют разработчики программ.

Оценку результатов испытания производит комиссия при активном участии разработчиков.

Независимые испытания проводят специальные подразделения, не несущие ответственности за разработку программ и непосредственно не подчиняющиеся руководителям разработки.

Технологическая схема испытания

Цель повышения эффективности испытания, его ускорения и удешевления может быть достигнута путем разработки технологической схемы испытаний, предусматривающей:

- знание назначения, испытываемого ПП, условий его функционирования и требований к нему со стороны пользователей;

- автоматизацию наиболее трудоемких процессов и, прежде всего, моделирование среды функционирования, включая искажающие воздействия;

- ясное представление цели и последовательности испытания;

- целенаправленность и не избыточность испытания, исключающая или минимизирующая повторение однородных процедур при одних и тех же условиях функционирования испытываемого ПП;

- систематический контроль, регулярное ведение протокола и журнала испытания;

- последовательное определение и выполнение плана испытания;

- сопоставление имеющихся ресурсов с предполагаемым объемом испытания;

- возможность обеспечения объективной количественной оценки полноты и достоверности результатов испытания на всех этапах.

Любому виду испытаний должна предшествовать тщательная подготовка.

В подготовку испытаний ПП входят следующие мероприятия:

- составление и согласование плана-графика проведения испытания;

- разработка, комплектование, испытание и паспортизация программно-технических средств, используемых при испытаниях;

- анализ пригодности испытательных средств, используемых во время предварительных испытаний, для проведения приемочных испытаний;

- анализ пригодности накопленных данных о качестве ПП для использования при окончательном определении значений показателей качества испытываемого ПП;

• проверка и согласование с представителем заказчика конструкторской документации на ПП, предъявляемой при испытаниях;

• разработка, согласование и утверждение программ и методик испытаний;

• аттестация специалистов на допуск к проведению испытаний;

• приемка испытываемого опытного образца ПП на носителе данных и документации;

• проведение мероприятий, направленных на обеспечение достоверности испытаний.

Следует подчеркнуть необходимость заблаговременной разработки и испытания программно-технических средств, которые будут использоваться при проведении испытаний.

При этом следует иметь в виду, что уровень точности и надежности измерительной аппаратуры должен быть значительно выше соответствующих показателей испытываемого объекта.

На основании изложенного можно определить следующие пять этапов испытания:

1. Обследование проектируемого ПП, анализ проектной документации.

2. Определение наиболее важных подсистем и функций проектируемого ПП, подлежащих испытанию.

3. Анализ показателей качества ПП и методов определения их значений. Разработка программ и методик испытания.

4. Разработка (освоение) испытательных программно-технических средств, библиотек тестов и баз данных (если они требуются).

5. Непосредственное проведение испытаний, анализ результатов, принятие решения.

В зависимости от специфики, условий применения, требований к качеству испытываемых ПП, испытания могут проводиться либо путем тестирования, либо путем статистического моделирования среды функционирования, либо на основе натурных и смешанных экспериментов. Часто полезно использование всех этих методов. Значения некоторых показателей качества могут быть получены экспертным путем.

На рис. 18 изображена технологическая схема в виде этапов подготовки и проведения испытания и их связи с этапами разработки ПП.

Планирование и оценка испытаний

План проведения испытаний должен быть ориентирован на обеспечение всесторонней проверки ПП и заданной достоверности полученных результатов при использовании ограниченных ресурсов, выделенных на испытания. Возможны следующие подходы для решения этой задачи:

1) анализируют диапазон входных данных. На основе анализа готовят множество комбинаций тестовых наборов данных, охватывающее наиболее характерные подмножества входных данных. Программу рассматривают как черный ящик. Испытания сводятся к последовательному вводу тестовых наборов данных и анализу получаемых результатов;

Рис. 18. Технологическая схема испытания ПП

2) анализируют множество ситуаций, которые могут возникнуть при функционировании ПП. Выбирают наиболее характерные. Каждую из них выражают через тестовый набор входных данных. Далее процесс испытания и анализа результатов сводится к п. 1;

3) с помощью граф-схемы анализируют микроструктуру ПП. Выбирают множество путей, которое полностью покрывает граф-схему, и такую последовательность тестовых наборов исходных данных, выполнение которой будет проходить по выделенным путям. Организация испытаний аналогична пунктам 1 и 2;

4) ПП испытывают в реальной среде функционирования;

5) ПП испытывают в статистически моделируемой среде функционирования, адекватной реальной среде.

Ни один из этих подходов не является универсальным. Каждый имеет свои преимущества и недостатки в зависимости от специфики испытываемого ПП. Например, подход 1 может оказаться предпочтительным, если диапазон входных данных обозрим, сравнительно легко анализируется и систематизируется, и неприемлемым - в противном случае. Наиболее достоверные результаты получаются при испытаниях в реальной среде функционирования. Но такие испытания редко удается осуществить. Поэтому на практике используют комбинации всех видов. Например, смешанный метод, когда среда функционирования ПП моделируется, а достоверность результатов проверяется сравнением с результатами, полученными при функционировании ПП в реальной среде.

Методика решения задачи планирования испытания включает в себя следующие этапы:

- нахождение всех путей реализации;

- выделение минимального подмножества путей, обеспечивающих проверку всех участков программы;

- разработка тестов для проверки выделенных путей.

Критерий интенсивности обнаружения ошибок. Если считать, что во время одного эксперимента обнаруживаются не более одной ошибки и каждая ошибка до начала следующего эксперимента устраняется, то можно предположить, что при благоприятном ходе отладки и испытания значение критерия интенсивности обнаружения ошибок N можно вычислить по формуле

$$N = 1 - n/K,$$

где n -	количество обнаруженных и устраненных ошибок;
K -	количество экспериментов.

С возрастанием количества экспериментов критерий интенсивности обнаружения ошибок будет асимптотически стремиться к единице.

Тогда в качестве критерия прекращения испытаний можно принять, например, следующее условие: $N > 0{,}95$ при обнаружении в последних двухстах экспериментах не более трех несущественных ошибок. Идея выбора такого критерия основана на том, что частота обнаружения ошибок, выраженная отношением n/K, по мере увеличения количества экспериментов должна уменьшаться и к моменту завершения испытаний принять значение, близкое к нулю. Следует иметь в виду, что оценка уровня завершенности испытаний будет достоверной лишь в случае, если каждый эксперимент проводится в новых условиях и испытатели стремятся обнаружить ошибки, а не доказать их отсутствие.

Критерий заданного значения средней наработки на отказ (критерий Дж.Д. Муса). Предположим, что суммарное количество обнаруженных и устраненных дефектов в программе n (под дефектом понимается любая причина неудовлетворенности свойствами программы) описывается показательной функцией времени функционирования t

$$n = N_0\left[1 - \exp\left(-\frac{Ct}{M_0 T_0}\right)\right],$$

где N_0 -	исходное количество дефектов в программе; ъ
M_0 -	общее количество дефектов, которое может проявиться за время эксплуатации ПП;
T_0 -	средняя наработка на отказ в начале испытаний;
C -	коэффициент сжатия тестов.

Коэффициент C≠ 1 тогда, когда абсолютная реактивность программы при прогоне тестов или статистических испытаниях отличается от абсолютной реактивности при работе программы в реальных условиях.

Если, например, за один час испытаний моделируется управляемый процесс, происходящий в реальных условиях в течение десяти часов, то коэффициент сжатия C принимается равным 10. Скорость обнаружения и устранения дефектов, измеряемая относительно времени функционирования программы, пропорциональна интенсивности отказов. Количество зарегистрированных отказов m зависит от суммарного времени функционирования программы следующим образом:

$$m = M_0\left[1-\exp\left(-\frac{Ct}{M_0 T_0}\right)\right].$$

Коэффициент пропорциональности B = n/m называется коэффициентом уменьшения дефектов.

Если в ходе испытания обнаруженные ошибки устраняются, то текущее значение средней наработки на отказ будет увеличиваться.

В качестве критерия завершенности испытания можно принять достижение требуемого (заданного) значения средней наработки на отказ T_0.

При планировании отладки и испытания программного обеспечения следует учитывать влияние следующих факторов:

- скорость выявления дефектов;
- скорость устранения дефектов;
- удовлетворенность машинным временем.

Первый фактор зависит от укомплектованности и квалификации испытателей, второй - от укомплектованности и квалификации группы программистов отладчиков, третий - от технической оснащенности разрабатывающей (испытывающей) организации.