Тестирование программных средств

- Основные определения
- Методы тестирования
- Стратегии тестирования
- Этапы тестирования
- Отладка программного средства
- Испытание программных продуктов

ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ ПО

Тестирование программного средства (ПС) - это процесс выполнения программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется тестовым или просто тестом. Тестирование программ является одной из составных частей более общего понятия - «отладка программ». Под отладкой понимается процесс, позволяющий получить программу, функционирующую с требующимися характеристиками в заданной области изменения входных данных.

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 ПС.

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

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

Тестирование - это процесс многократного выполнения программы с целью выявления ошибок. Целью тестирования является обнаружение максимального числа ошибок.

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

1. Процесс тестирования более эффективен, если проводится не автором программы.

2. Необходимой частью тестового набора данных должно быть описание предполагаемых значений результатов тестовых прогонов. 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.

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.

5. Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и не делает ли она того, чего не должна делать. 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.

Основные определения

Тестирование (testing) - процесс выполнения программы или ее части с целью найти ошибки.

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.

Контроль (verification) - попытка найти ошибки, выполняя программу в тестовой, или моделируемой, среде.

Испытание (validation) - попытка найти ошибки, выполняя программу в заданной реальной среде.

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

Отладка (debugging) не является разновидностью тестирования. Слова «отладка» и «тестирование» часто используются как синонимы

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 модуля иногда включает математическое доказательство.

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.

Стратегия проектирования тестов

В тестирование ПО входят постановка задачи для теста, проектирование, написание тестов, тестирование тестов, выполнение тестов и изучение результатов тестирования. Важную роль играет проектирование теста. Возможны следующие подходы к стратегии проектирования тестов:

1. Тестирование по отношению к спецификациям (не заботясь о тексте программы).

2. Тестирование по отношению к тексту программы (не заботясь о спецификациях).

Интеграция модулей

Вторым по важности аспектом тестирования (после проектирования тестов) является последовательность слияния всех модулей в систему или программу. Выбор этой последовательности (должен приниматься на уровне проекта и на ранней стадии) определяет форму, в которую записываются тесты, типы необходимых инструментов тестирования, последовательность программирования модулей, тщательность и экономичность всего этапа тестирования.

Существует несколько подходов, которые могут быть использованы для слияния модулей в более крупные единицы.

Методы тестирования

В практике тестирования используются методы: статический, детерминированный, стохастический и в реальном масштабе времени.

Статическое тестирование - проводится путем просмотра текста программы, проверки правил структурного построения программ и обработки данных. В качестве эталонов используются, во-первых, внутренние спецификации, а, во-вторых, коллективный опыт специалистов-тестировщиков. Применение статического тестирования достаточно эффективно. Для типичных программ, по данным фирмы IBM, можно находить от 30 % до 80 % ошибок логического проектирования и кодирования.

Детерминированное тестирование - это многократное выполнение программы с использованием определенных, специальным образом подобранных тестовых наборов данных. При детерминированном тестировании контролируются каждая комбинация исходных данных и соответствующие результаты, а также каждое утверждение в спецификации тестируемой программы. Этот метод наиболее трудоемкий.

Стохастическое тестирование предполагает использование в качестве исходных данных множество случайных величин с соответствующими распределениями. Для сравнения полученных результатов используются также распределения случайных величин. Стохастическое тестирование применяется в основном для обнаружения ошибок, а для диагностики и локализации ошибок приходится переходить к детерминированному тестированию с использованием конкретных значений исходных данных, из области изменения ранее использовавшихся случайных величин. Стохастическое тестирование наилучшим образом подвергается автоматизации путем использования датчиков случайных чисел (генераторов случайных величин) и применяется для комплексного тестирования ППП.

Тестирование в реальном масштабе времени осуществляется для ППП, предназначенных для работы в системах реального времени. В процессе такого тестирования проверяются результаты обработки исходных данных с учетом времени их поступления, длительности и приоритетности обработки, динамики использования памяти и взаимодействия с другими программами. При обнаружении отклонения результатов выполнения программ от ожидаемых для локализации ошибок, приходится фиксировать время и переходить к детерминированному тестированию.

Детерминированное тестирование основывается на двух подходах: структурное тестирование и функциональное тестирование

стратегии белого ящика:

- покрытие операторов ;
- покрытие узлов ветвления (покрытие решений) ;
- покрытие условий;
- комбинаторное покрытие условий/решений.

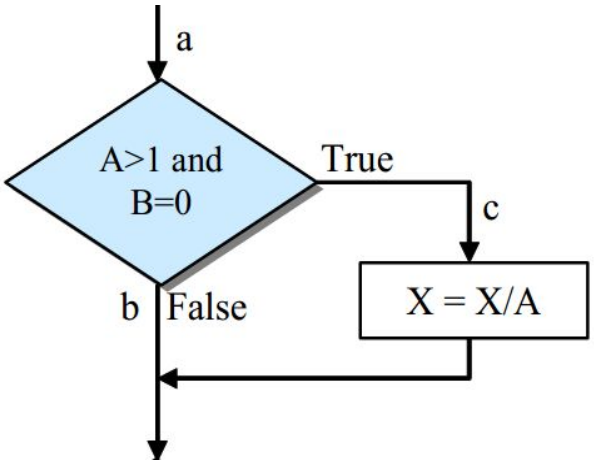
стратегии черного ящика:

- эквивалентное разбиение;
- анализ граничных значений ;
- применение функциональных диаграмм;

метод покрытия операторов требует выполнение каждого оператора программы, по крайней мере, один раз.

Покрытие решений - каждое направление перехода должно быть реализовано по крайней мере один раз.

покрытие условий. В этом случае записывают число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.



A > 1, A ≤ l, B = 0 и B ≠ 0

Тест

1. A = 2, B = 0
2. A = 1, B = 1

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

1. A > 1, B = 0.
2. A > 1, B ≠ 0.
3. A ≤ 1, B = 0.
4. A ≤ l, B ≠ 0.

Тест

A = 2, B = 0,

A = 2, B = 1;

A = 1, B = 0;

A = 1, B = 1.

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа:

1) выделение классов эквивалентности;
2) построение тестов .

Классы эквивалентности выделяются путем выбора каждого входного условия (обычно это предложение или фраза в спецификации) и разбиением его на две или более групп.

различают два типа классов эквивалентности: правильные классы эквивалентности , представляющие правильные входные данные программы , и неправильные классы эквивалентности , представляющие все другие возможные состояния условий (т. е. ошибочные входные значения).

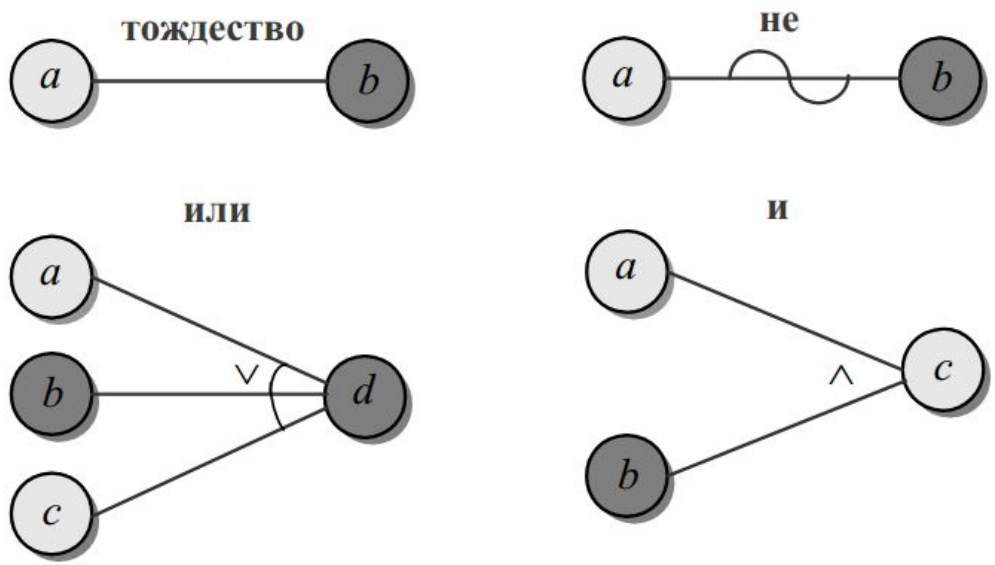
Граничные условия – это ситуации , возникающие непосредственно на , выше или ниже

границ входных и выходных классов эквивалентности.

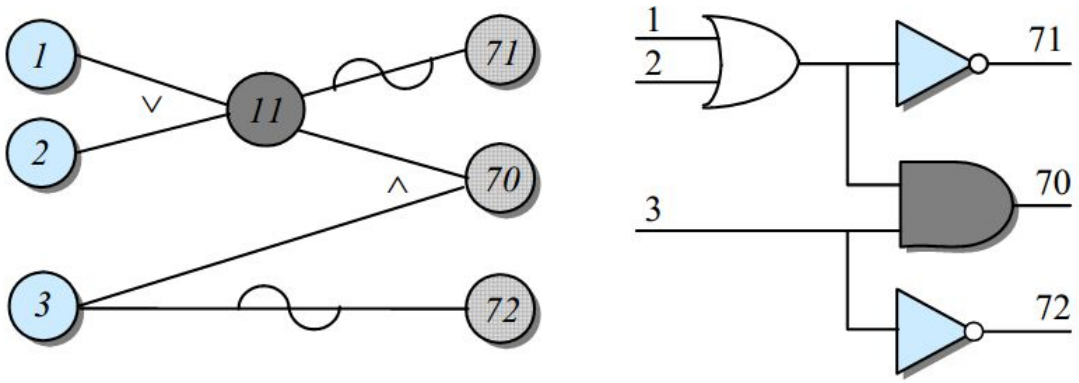
Функциональная диаграмма представляет собой формальный язык, на который транслируется спецификация, написанная на естественном языке.

Построение тестов этим методом осуществляется в несколько этапов .

1. Спецификация разбивается на « рабочие» участки.
2. В спецификации определяются причины и следствия. Причина есть отдельное входное условие или класс эквивалентности входных условий . Следствие есть выходное условие или преобразование системы (остаточное действие , которое входное условие оказывает на состояние программы или системы). Каждым причине и следствию приписывается отдельный номер .
3. Анализируется семантическое содержание спецификации, которая преобразуется в булевский граф , связывающий причины и следствия. Это и есть функциональная диаграмма.
4. Диаграмма снабжается примечаниями, задающими ограничения и описывающими комбинации причин и (или) следствий, которые являются невозможными из -за синтаксических или внешних ограничений.
5. Путем методического прослеживания состояний условий диаграммы она преобразуется в таблицу решений с ограниченными входами . Каждый столбец таблицы решений соответствует тесту.
6. Столбцы таблицы решений преобразуются в тесты.

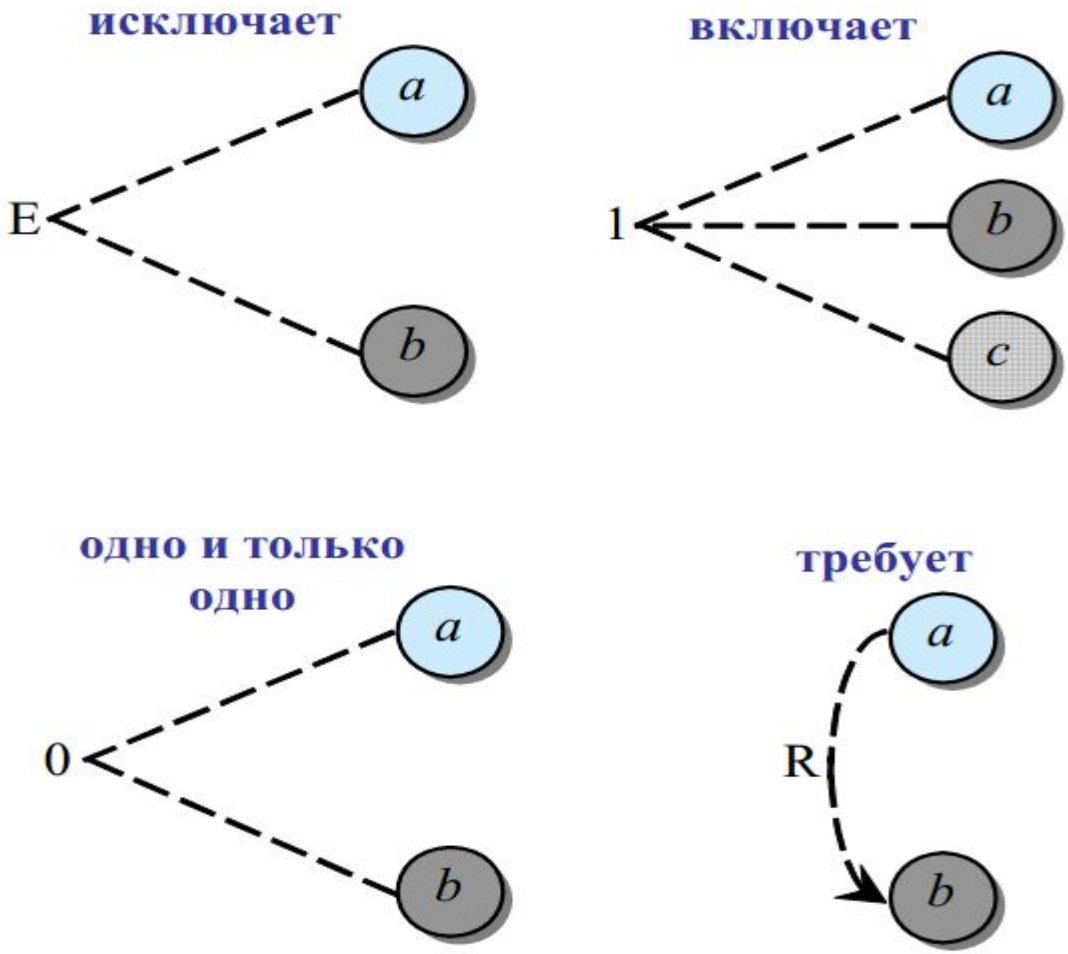


Базовые логические отношения функциональных диаграмм



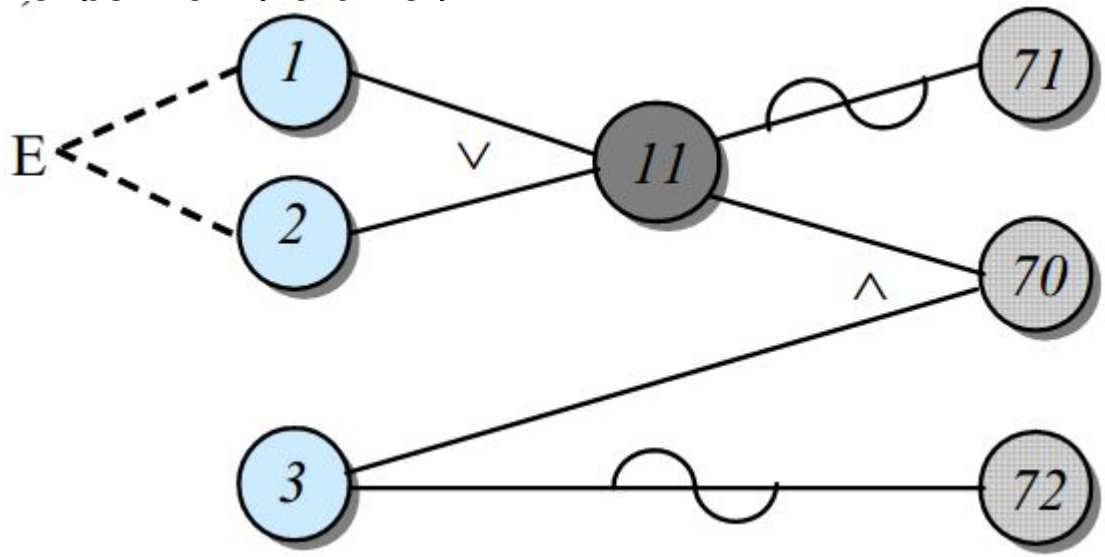
Пример функциональной диаграммы и эквивалентной логической схемы

причины 1 и 2 не могут быть установлены в 1 одновременно
В этом случае используются дополнительные логические ограничения



Символы ограничений

Ограничение E устанавливает, что E должно быть истинным, если хотя бы одна из величин – a или b – принимает значение 1 (a и b не могут принимать значение 1 одновременно). Ограничение I устанавливает, что, по крайней мере, одна из величин a , b или c всегда должна быть равной 1 (a , b и c не могут принимать значение 0 одновременно). Ограничение 0 устанавливает, что одна и только одна из величин a или b должна быть равна 1. Ограничение R устанавливает, что если a принимает значение 1, то и b должна принимать значение 1 (т . е. невозможно, чтобы a была равна 1, а b – 0).



Восходящее тестирование

При восходящем подходе программа собирается и тестируется снизу вверх. Только модули самого нижнего уровня («терминальные» модули; модули, не вызывающие других модулей) тестируются изолированно, автономно. После того как тестирование этих модулей завершено, вызов их должен быть так же надежен, как вызов встроенной функции языка или оператор присваивания. Затем тестируются модули, непосредственно вызывающие уже проверенные. Эти модули более высокого уровня тестируются не автономно, а вместе с уже проверенными модулями более низкого уровня. Процесс повторяется до тех пор, пока не будет достигнута вершина. Здесь завершаются и тестирование модулей, и тестирование сопряжений программы.

При восходящем тестировании для каждого модуля необходим драйвер: нужно подавать тесты в соответствии с сопряжением тестируемого модуля. Одно из возможных решений - написать для каждого модуля небольшую ведущую программу. Тестовые данные представляются как «встроенные» в эту программу переменные и структуры данных, и она многократно вызывает тестируемый модуль, с каждым вызовом передавая ему новые тестовые данные. Имеется и лучшее решение: воспользоваться программой тестирования модулей - это инструмент тестирования, позволяющий описывать тесты на специальном языке и избавляющий от необходимости писать драйверы.

Нисходящее тестирование

При нисходящем подходе программа собирается и тестируется сверху вниз. Изолировано тестируется только головной модуль. После того как тестирование этого модуля завершено, с ним соединяются (например, редактором связей) один за другим модули, непосредственно вызываемые им, и тестируется полученная комбинация. Процесс повторяется до тех пор, пока не будут собраны и проверены все модули. Для имитации функций недостающих модулей программируются модули-заглушки, которые моделируют функции отсутствующих модулей.

Метод большого скачка

Один из подходов к интеграции модулей - это метод большого скачка. В соответствии с этим методом каждый модуль тестируется автономно. По окончании тестирования модулей они интегрируются в систему все сразу.

Метод сандвича

Это компромисс между восходящим и нисходящим подходами.

При использовании этого метода одновременно начинают восходящее и нисходящее тестирование, собирая программу как снизу, так и сверху и встречаясь, где-то в середине.

Этапы тестирования

Процесс тестирования ППП начинается проверкой правильности работы отдельных модулей и заканчивается приемкой после испытания ППП при его сдаче заказчику или началом коммерческих продаж ППП. Рассмотрим типичные этапы работы тестировщиков.

Тестирование программных модулей - наиболее формализованный и автоматизированный процесс тестирования.

Основная задача тестирования состоит в проверке обработки программными модулями поступающей информации и корректности, получающихся на выходе данных в соответствии с функциями, отраженными в спецификациях.

Проверяется корректность структуры модулей и их конструктивных основных компонентов: процедур, циклов, блоков, условий и т.д.

Тестирование планируется с учетом структуры модулей и особенностей обработки информации и осуществляется преимущественно детерминировано.

Тестирование функциональных групп модулей предназначено для проверки корректности решения крупных автономных Функциональных задач ППП. Проверяется правильность управляющих и информационных связей между модулями, а также корректность вычислений в процессе обработки информации.

Детерминированным тестированием проверяются структура групп программ и основные маршруты обработки информации.

Комплексное тестирование - сложный процесс, в котором завершается проверка корректности функционирования программ при правильных исходных данных, и осуществляются основные проверки при искажениях на входе.

Проверяются надежность функционирования всего ППП в реальных условиях, эффективность средств программой защиты и восстановления. Определяются корректность использования программами ресурсов компьютера и функционирование программ в критических условиях. Формализация процесса тестирования на этом этапе наиболее трудна, и оценка полноты тестирования осуществляется преимущественно по степени выполнения функций и по характеристикам надежности функционирования ППП. Для этого применяются преимущественно стохастическое тестирование и тестирование в реальном времени.

Системное тестирование (или испытание программного комплекса) предназначено в основном для проверки соответствия пакета прикладных программ техническому заданию и для оценки его пригодности к регулярной эксплуатации и сопровождению.

Для этого проверяются полнота и точность технической документации, качество функционирования пакета прикладных программ по всем требованиям технического задания.

Проверка пригодности к сопровождению включает тестирование настройки версий на условия конкретного применения и анализ удобства модифицирования версий пакета прикладных программ.

Отладка программного средства.

Отладка программного средства (ПС) - это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ.

Отладка = Тестирование + Поиск ошибок + Редактирование

Комплексная отладка программного средства

Тестирование архитектуры ПС. Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ ПС.

Тестирование внешних функций. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ ПС.

Тестирование качества ПС. Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества ПС. Это наиболее трудный и наименее изученный вид тестирования.

Тестирование документации по применению ПС. Целью тестирования является поиск несогласованности документации по применению и совокупностью программ ПС, а также выявление неудобств, возникающих при применении ПС.

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.

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.

На основании изложенного можно определить следующие пять этапов испытания:

1. Обследование проектируемого ПС, анализ проектной документации.

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

3. Анализ показателей качества ПС 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.