1. **ФАКТОРЫ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**
2. Функциональность (Functionality)

* Совместимость в смысле алгоритма
* Точность (соответствие между функцией и требованиями)
* Взаимодействие
* Безопасность

1. Надежность (Reliability)

* Зрелость
* Толерантность к ошибке
* Самовосстановление

1. Удобство использования (Usability)

* Способность к пониманию (understandability)
* Способность к обучению (learnability)
* Работоспособность (operability)
* Привлекательность (attractiveness)

1. Эффективность (Efficiency)

* Временное поведение (time behavior)
* Использование ресурсов

1. Сопровождаемость (Maintainability)

* Анализируемость (Analysability)
* Изменяемость (Changeability)
* Стабильность (Stability)
* Тестируемость (Testability)

6. Мобильность (Portability)

* Адаптируемость (Adaptability)
* Настраиваемость (простота внедрения, Installability)
* Совместимость (Co-existence)
* Взаимозаменяемость (Replaceability)

1. **МЕТРИКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ХОЛСТЕДА**

|  |  |
| --- | --- |
| **Метрика** | **Краткое описание** |
| *n1* | число уникальных операторов |
| *N1* | общее число всех операторов |
| *n2* | число уникальных операндов |
| *N2* | общее число всех операндов |
| *n* | *n = n1 + n2* – словарь программы |
| *N* | *N = N1 + N2* – длина программы |
| *N'* | – теоретическая длина программы |
| *V* | *V = N \* log2(n)* – объем программы |
| *V'* | *V' = N' \* log2(n')* – теоретический объем программы |
| *n’* | количество переменных, которые вводятся в программу + количество переменных, которые программа выводит |
| *L* | *L = V' / V* – уровень качества программы |
| *λ* | *λ = L / V'* – уровень языка программирования |
| *D* | *D =* ***1*** */ L'* – трудоемкость кодирования программы |
| *I* | *I = V / D* – информационное содержание программы, данная характеристика позволяет определить умственные затраты на создание программы |
| *E* | *E = V / L* – необходимые интеллектуальные усилия |
| *T* | *T = E / S* – время составления программы, (*S = 5..20*) |
| *T'* | оценка времени составления программы |
| *B* | *B = E2/3 / 3000* – число дефектов в программе |
| *B'* | *B' = V / 3000* – оценка числа дефектов в программе |
| *Mх* | *Mх = n’ / 6* – М Холстеда (количество модулей) |
| *Мм* | Цикломатическое число Маккейба. Способы нахождения:   1. На диаграмме деятельности количество ребер – количество узлов (блоков) + 2; 2. Количество условных операторов (if, else, else if, for, switch, while, do..while) + 1. Вложенные условные конструкции нужно также считать (например, **for** внутри **for**). Конструкция **if..else** считается как 1. |

1. **ОПРЕДЕЛЕНИЕ ПОНЯТИЯ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Верификацией и аттестацией называются процессы проверки и анализа, в ходе которых проверяется соответствие программного обеспечения своей спецификации и требованиям заказчиков. Верификация и аттестация охватывают весь цикл жизни ПО.

1. **ОБНАРУЖЕНИЕ ОШИБОК**

**Пассивное обнаружение**. Меры по обнаружению ошибок могут быть приняты на нескольких структурных уровнях программной системы.

**Активное обнаружение ошибок**. Не все ошибки можно выя­вить пассивными методами, поскольку эти методы обнаружива­ют ошибку лишь тогда, когда ее симптомы подвергаются соот­ветствующей проверке.

1. **КОНЦЕПЦИЯ ТЕСТИРОВАНИЯ**

Концепция тестирования должна отвечать, как минимум на следующие вопросы:

1. Какие наборы тестов существую в проекте, по каким принципам они формируются и в какие моменты выполняются?
2. Как проводится тестирование багов, создаются ли на них тесты, в какой набор тестов попадают?
3. Как проводится тестирование новых фитч, какие тесты на них создаются, в какой набор тестов попадают?

Да, и надо помнить про двойственность проекта.

1. **СВЯЗЬ ЗАДАЧ ВАЛИДАЦИИ, ВЕРИФИКАЦИИ И ТЕСТИРОВАНИЯ С ЖИЗНЕННЫМ ЦИКЛОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**Валидация** программного обеспечения - процесс определения соответствия разрабатываемого программного обеспечения между ожиданиями и потребностями пользователя, требованиям к системе.

**Верификация и валидация** заключается в проверке спецификаций и правильности выполнения программ в соответствии с заданными требованиями и формального описания программы.

**Верификация** программного кода помогает сделать вывод о корректности созданной программной системы при ее проектировании и после завершения ее разработки.

1. **СТАДИИ ТЕСТИРОВАНИЯ В ПРОЦЕССЕ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**





1. **МОДЕЛЬ УПРАВЛЕНИЯ КАЧЕСТВОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ – CMMI**

**Комплексная модель производительности и зрелости (CMMI)** – набор моделей (методологий) совершенствования процессов в организациях разных размеров и видов деятельности.

**CMMI** содержит набор рекомендаций в виде практик, реализация которых, по мнению разработчиков модели, позволяет реализовать цели, необходимые для полной реализации определённых областей деятельности.

1. **ОСНОВНЫЕ ПРОБЛЕМЫ ПРОЦЕССА ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

В прикладном, узкоспециализированном ПО основной проблемой тестирования является сложность с имитировать поведение человека в тесте.

1. **МОДУЛЬНОЕ ТЕСТИРОВАНИЕ**

Тестирование программного кода на уровне отдельно взятого модуля, функции или класса. Основная цель – найти ошибку в алгоритме. Модульное тестирование проводится по принципу «Белого ящика».

1. **ТЕСТИРОВАНИЕ МЕТОДОМ "БЕЛЫЙ ЯЩИК"**

**Тестирование методом белого ящика** (также: прозрачного, открытого, стеклянного ящика; основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки.

1. **НАДЕЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Способность программного продукта поддерживать заданный уровень качества функционирования при его использовании в заданных условиях.

1. **СИСТЕМНОЕ ТЕСТИРОВАНИЕ**

Основной задачей **системного тестирования** является проверка как функциональных, так и не функциональных требований в системе в целом. При этом выявляются **дефекты**, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д.

1. **МОДЕЛЬ КОМПЛЕКСНОГО УПРАВЛЕНИЯ КАЧЕСТВОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (НА БАЗЕ ISO)**

Комплексное решение задачи обеспечения качества программных систем предполагает разработку и внедрение той или иной системы управления качеством.

Уровнями модели будут выступать:

* Факторы – первый уровень.
* Характеристики – второй уровень.
* Метрики – третий уровень.
* Оценочные элементы – четвертый уровень.

1. **МОДЕЛЬ ИЗМЕРЕНИЯ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**Качество** – это характеристика которая поддается воздействию.

**Характеристика качества** – отображает свойства ПО, по которым можно судить что оно качественное (по которым на него можно воздействовать). Поскольку очень сложно измерить, доказать качество, то вводят иерархические системы измерения качества. Каждый программный продукт описывается своей уникальной системой качества, а представление модели ISO является общей.

**Фактор качества** – это свойства, в той или иной степени описывающие качество ПО. При оценке качества учитывается 2,3 фактора. Для численной оценки качества используют 1 или несколько критериев.

**Критерий качества** – это вычислимое выражение, которое устанавливает расстояния (интервал) по которым мы можем сделать вывод о качестве ПО с точки зрения заданного фактора.

1. **КЛАССИФИКАЦИЯ МЕТОДОВ ВЕРИФИКАЦИИ**

**Экспертиза**. В качестве видов экспертиз традиционно выделяют организационные экспертизы, технические экспертизы, сквозной контроль, инспекции и аудиты.

**Статический анализ.** Статический анализ используется для поиска часто встречающихся ошибок по некоторым шаблонам.

**Формальные методы верификации**. Формальные методы верификации используют для анализа свойств ПО.

**Динамические методы верификации.** Динамические методы верификации, в рамках которых анализ и оценка свойств программной системы делаются по результатам ее реальной работы или работы некоторых ее моделей.

**Синтетические методы верификации.** В последние время появилось множество инструментальных средств, в которых применяются элементы нескольких перечисленных выше видов верификации.

1. **МУТАЦИОННЫЕ КРИТЕРИИ ВЫБОРА ТЕСТОВ**

В основе их применения – будем считать, что программист пишет с 1-го раза почти правильный программный код. Назовем **мутациями** – мелкие ошибки. **Мутанты** – те программы, которые отличаются **мутациями**. Метод означает, что в разрабатываемом программном коде Р, намеренно вносят мутации, формируя набор программного кода с ошибками Р1, …, Рm. Если все тестовые наборы заблокировали копии программного кода, то программа Р – протестирована.

1. **РОЛИ В ПРОЦЕССЕ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

1 роль – **Заказчик**: имеет право менять требования лишь во взаимодействии с менеджером или лидером проекта.

2 роль – **Менеджер проекта**: обеспечивает коммуникационный канал между заказчиком и командой. Основная обязанность – получить требования.

3 роль – **Менеджер ПО:** управляет командой внутри проекта (коммуникатор). Отвечает за разработку функциональных спецификаций. Может их менять, распределять разработчикам задачи и отчитываться об этапах.

4 роль – **Разработчик.**

5 роль – **Специалист по тестированию**: разработка тест планов по тестированию.

6 роль – **Специалист по контролю качества**: отвечает за альфа и бета тестирование.

7 роль – **Специалист по верификации.**

8 роль – **Специалист по внедрению и сопровождению.**

9 роль – **Специалист по безопасности.**

10 роль – **Технический писатель.**

1. **КОЛИЧЕСТВЕННЫЕ ХАРАКТЕРИСТИКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ЕГО НАДЕЖНОСТИ**

* **Вероятность безотказной работы**
* **Вероятность отказа**
* **Интенсивность отказов ПО системы**
* **Средняя наработка до отказа**
* **Среднее время восстановления**
* **Коэффициент готовности**

1. **ФУНКЦИОНАЛЬНЫЕ КРИТЕРИИ ВЫБОРА ТЕСТОВ**

**Функциональные критерии** – контролируют степень выполнения требований заказчика к ПО. **Функциональные критерии** работают по модели черного ящика (неизвестно что внутри, манипуляции только со входами и выходами):

1. **Тестирование пунктов спецификации** (создается тест который проверяет каждый пункт спецификации хотя бы 1 раз).
2. **Тестирование классов входных данных** (проверить каждый класс не менее 1 раза).
3. **Разработка тестовых наборов на основе классов эквивалентности.**
4. **Тестирование правил** (в качестве правила рассматриваем соответствие входных и выходных правил либо вводится спец-я грамматика).
5. **Тестирование классов выходных данных.**
6. **Тестирование функций** (это самый популярный критерий по кот мы создаем тестовые наборы, чтобы проверить хотя бы 1 раз любую из функций представленных на UML-диаграмме).
7. **КЛАССИФИКАЦИЯ ПРОГРАММНЫХ ОШИБОК**
8. **Сбой** – кратковременный отказ.
9. **Дефект** – долговременный отказ.
10. **Ошибка** – в ПО это неверный, неправильный перевод из одного представление в другое.
11. **Отказ** – это отклонение программы от функционирования или невозможность программы выполнять функции, определенные требованиями и ограничениями.
12. **СТОХАСТИЧЕСКИЕ КРИТЕРИИ ВЫБОРА ТЕСТОВ**

Существует **2 критерия** на основе этой процедуры:

* Статистический метод окончания тестирования. В основе лежит предположение о соблюдении гипотез закона распределения генерируемых данных и тестового набора.
* Метод оценки скорости выявления ошибки. Этот критерий оценивает интервал времени между выявлением текущей ошибки и последующей.

1. **ОСОБЕННОСТИ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ ДЛЯ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**

Для того что бы выполнить работу по тестированию необходимо выявить 2 способа связи между объектами в тестируемой иерархии.

1. Прямой вызов (на прямую вызвать 1 метод из другого Р-пути)
2. Обработка сообщений (ММ – путь)

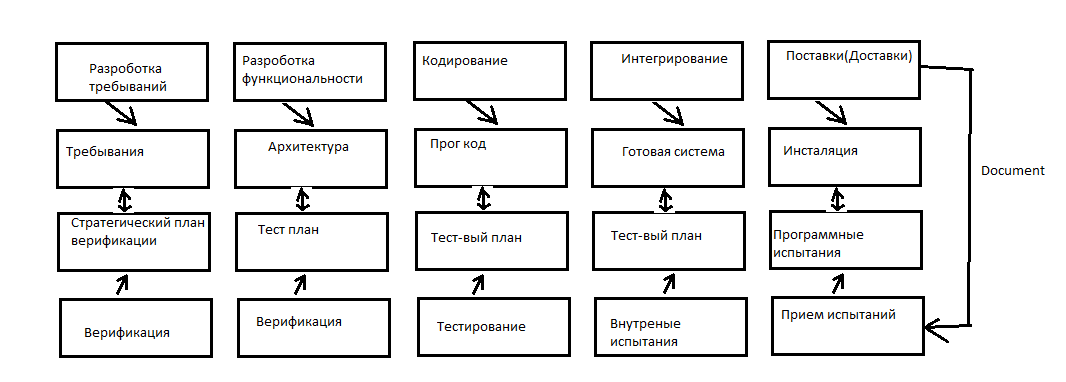
Р-путь возникает в случае вызова командной строки, код обрабатывает алгоритм и выгружается из памяти. При этом оно может создать ММ-путь.

1. **СТРУКТУРНЫЕ КРИТЕРИИ ВЫБОРА ТЕСТОВ**

Существует 3 основных критерия:

1. С0 – **тестирование команд** – это означает что необходимо создать множество тестовых наборов. При выполнении кот каждый оператор срабатывает хотя бы 1 раз.
2. С1 – **тестирование ветвей** – создать набор тестов кот обеспечивает прохождения каждой ветки алгоритма минимум 1 раз.
3. С2 – **тестирование путей** – создать набор тестов обеспечивающих прохождения путей минимум 1 раз.
4. **ДОКУМЕНТИРОВАНИЯ В ПРОЦЕССЕ ВЕРИФИКАЦИИ**

Жизненный цикл



1. **ВИЗНАЧЕННЯ ЯКОСТІ ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ (ISO, IEEE)**
2. Функциональность (Functionality)

* Совместимость в смысле алгоритма
* Точность (соответствие между функцией и требованиями)
* Взаимодействие
* Безопасность

1. Надежность (Reliability)

* Зрелость
* Толерантность к ошибке
* Самовосстановление

1. Удобство использования (Usability)

* Способность к пониманию (understandability)
* Способность к обучению (learnability)
* Работоспособность (operability)
* Привлекательность (attractiveness)

1. Эффективность (Efficiency)

* Временное поведение (time behavior)
* Использование ресурсов

1. Сопровождаемость (Maintainability)

* Анализируемость (Analysability)
* Изменяемость (Changeability)
* Стабильность (Stability)
* Тестируемость (Testability)

6. Мобильность (Portability)

* Адаптируемость (Adaptability)
* Настраиваемость (простота внедрения, Installability)
* Совместимость (Co-existence)
* Взаимозаменяемость (Replaceability)

1. **КЛЮЧЕВЫЕ ПРИНЦИПЫ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ**
2. Автоматизация тестирования **экономит время**.
3. Она **исключает человеческий фактор**.
4. Автоматизация тестирования **дает возможность работать без графического пользовательского интерфейса**.
5. Средства автоматизации **позволяют эмулировать многопользовательскую работу для нагрузочного тестирования**.
6. Практически все средства АТ **имеют инструментарий фиксации ошибок и результатов**.
7. **ТЕСТИРОВАНИЕ ТАБЛИЦЫ ПЕРЕХОДОВ СОСТОЯНИЙ**

**Тестирование таблицы переходов** — это метод тестирования «черного ящика», который используется там, где некоторый аспект системы может быть описан в так называемом «конечном автомате». Это просто означает, что система может находиться в (конечном) числе разных состояний, а переходы из одного состояния в другое определяются правилами «машины».

1. **НА ПРИМЕРЕ СИСТЕМЫ MANTIS ДАЙТЕ ХАРАКТЕРИСТИКИ СИСТЕМЕ ОТСЛЕЖИВАНИЯ ОШИБОК**

* Обеспечивает взаимодействие разработчиков с пользователями (тестировщиками).
* Позволяет пользователям заводить сообщения об ошибках и отслеживать дальнейший процесс работы над ними со стороны разработчиков.
* Система имеет гибкие возможности конфигурирования, что позволяет настраивать её не только для работы над программными продуктами, но и в качестве системы учёта заявок для Helpdesk.
* Возможна интеграция с wiki-движком для создания документации (DokuWiki).
* Система является веб-приложением, поэтому не требует для работы специального ПО на стороне клиента и работает через веб-браузер.

1. **НАЗНАЧЕНИЕ И ОСНОВНЫЕ КОМПОНЕНТЫ ОТЧЕТА ОБ ОШИБКЕ**

**Отчет об ошибке (Баг репорт)** – это технический документ, который содержит в себе полное описание бага, включающее информацию, как о самом баге (короткое описание, серьезность, приоритет и т.д.), так и о условиях возникновения данного бага.

В общем случае, баг репорт состоит из:

* **Шапка.**
* **Окружение.**
* **Описание.**
* **Дополнения.**
* **Краткое описание.**
* **Серьезность**.
* **Шаги к воспроизведению.**
* **Фактический результат.**
* **Ожидаемый результат.**