Тестиирование

По Swebok

Определение

* **Тестирование** (software testing) – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах.
* Тестирование программных систем состоит из *динамической* верификации поведения программ на *конечном (ограниченном)* наборе тестов (set of test cases), *выбранных* соответствующим образом из обычно выполняемых действий прикладной области и обеспечивающих проверку соответствия *ожидаемому* поведению системы.

Динамичность (dynamic)

* Термин подразумевает тестирование всегда предполагает выполнение тестируемой программы с заданными входными данными. При этом, величины, задаваемые на вход тестируемому программному обеспечению, не всегда достаточны для определения теста. Сложность и недетерминированность систем приводит к тому, что система может по разному реагировать на одни и те же входные параметры, в зависимости от состояния системы. В данной области знаний термин “вход” (input) будет использоваться в рамках соглашения о том, что вход может также специфицировать состояние системы, в тех случаях, когда это необходимо. Кроме динамических техник проверки качества, то есть тестирования, существуют также и статические техники, рассматриваемые в области знаний “Software Quality”.

Конечность (ограниченность, finite)

* Для простых программ теоретически возможно столь большое количество тестовых сценариев, что исчерпывающее тестирование может занять многие месяцы и даже годы. Именно поэтому, с практической точки зрения, всестороннее тестирование считается бесконечным. Тестирование всегда предполагает компромисс между ограниченными ресурсами и заданными сроками, с одной стороны, и практически неограниченными требованиями по тестированию, с другой. То есть мы снова говорим об определении характеристик “приемлемого” качества, на основе которых планируем необходимы объем тестирования.

Выбор (selection)

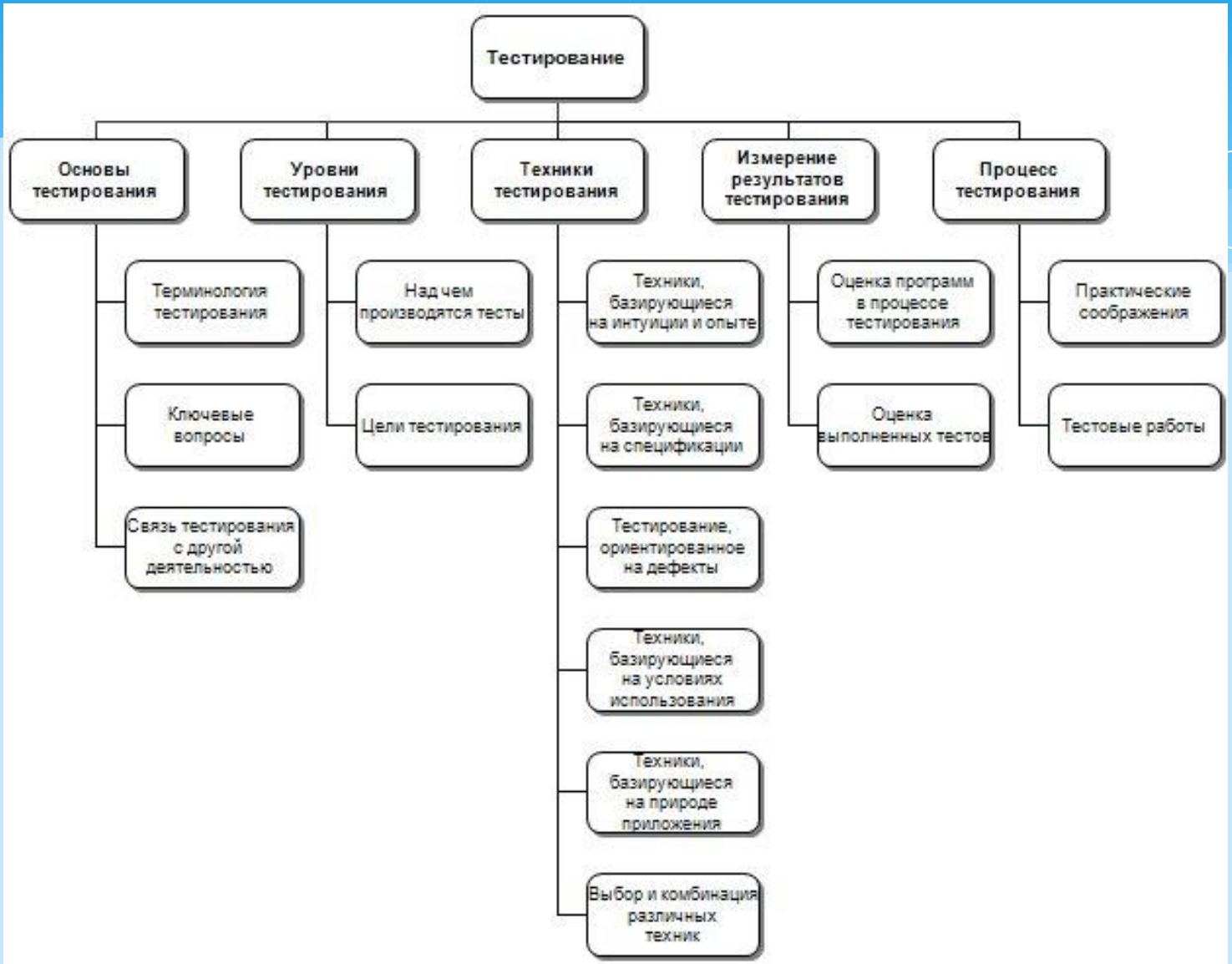
* Многие предлагаемые техники тестирования отличаются друг от друга в том, как выбираются сценарии тестирования. Инженеры по программному обеспечению должны обладать представлением о том, что различные критерии выбора тестов могут давать разные результаты, с точки зрения эффективности тестирования. Определение подходящего набора тестов для заданных условий является очень сложной проблемой. Обычно, для выбора соответствующих тестов совместно применяют техники анализа рисков, анализ требований и соответствующую экспертизу в области тестирования и заданной прикладной области.

Ожидаемое поведение (*expected* behaviour)

* Хотя это не всегда легко, все же необходимо решить, какое наблюдаемое поведение программы будет приемлемо, а какое – нет. В противном случае, усилия по тестированию – бесполезны. Наблюдаемое поведение может рассматриваться в контексте пользовательских ожиданий (подразумевая “тестирования для проверки” - testing for validation), спецификации (“тестирование для аттестации” - testing for verification) или, наконец, в контексте предсказанного поведения на основе неявных требований или обоснованных ожиданий. См. тему SWEBOK 6.4 “Приемочные тесты” области знаний “Software Requirements”.

*Качество программного обеспечения” (Software Quality)

* *статические (без выполнения кода)*
* *динамические (с выполнением кода)*


Тестирование
Основы тестирования
Уровни тестирования
Техники тестирования
Измерение результатов тестирования
Процесс тестирования
Терминология тестирования
Ключевые вопросы
Связь тестирования с другой деятельностью
Над чем производятся тесты
Цели тестирования
Техники, базирующиеся на интуиции и опыте
Техники, базирующиеся на спецификации
Тестирование, ориентированное на дефекты
Техники, базирующиеся на условиях использования
Техники, базирующиеся на природе приложения
Выбор и комбинация различных техник
Оценка программ в процессе тестирования
Оценка выполненных тестов
Практические соображения
Тестовые работы

1. Основы тестирования (Software Testing Fundamentals)

* Основные понятия в области тестирования, базовые термины, ключевые проблемы и их связь другими областями деятельности и знаний.

1.1 Терминология тестирования (Testing-Related Terminology)

* IEEE Standard 610-90 (Standard Glossary of Software Engineering Terminology)

1.1.2 Недостатки и сбои (Faults vs. Failures)

* Существующие термины: недостатки (faults), дефекты (defects), сбои (failures), ошибки (errors)
* SWEBOK:
* *причина* нарушения работы (*недостаток или дефект)*
* нежелательный эффект, вызываемый этими причинами – *сбой.*
* *ошибка*, в зависимости от контекста, может описывать и как причину сбоя, и сам сбой. **Тестирование позволяет обнаружить дефекты, приводящие к сбоям**

1.2 Ключевые вопросы (Key Issues)

1.2.1 Критерии отбора тестов/критерии адекватности тестов, правила прекращения тестирования (Test selection criteria/test adequacy criteria, or stopping rules)

* Критерии отбора тестов могут использоваться как для создания набора тестов, так и для проверки, насколько выбранные тесты адекватны решаемым задачам (тестирования). При этом, обсуждаемые критерии помогают определить, когда можно или необходимо прекратить тестирование.

1.2.2 Эффективность тестирования/Цели тестирования (Test effectiveness/Objectives for testing)

* **Тестирование** – это наблюдение за выполнением программы, запущенной в целях тестирования с заданными параметрами, по заданному сценарию или с другими заданными начальными условиями или целями тестирования.
* Эффективность теста может быть определена только в контексте заданных условий.

1.2.3 Тестирование для идентификации дефектов (Testing for defect identification)

* Случай тестирования подразумевает успешность процедуры тестирования, если дефект найден. Это отличается от подхода в тестировании, когда тесты запускаются для демонстрации того, что программное обеспечение удовлетворяет предъявляемым требованиями и, соответственно, тест считается успешным, если не найдено дефектов.

1.2.4 Проблема оракула (The oracle problem)

* “Оракул” - любой агент (человек или программа), оценивающий поведение программы, формулируя вердикт - тест пройден (“pass”) или нет (“fail”).

1.2.5 Теоретические и практические ограничения тестирования (Theoretical and practical limitation of testing)

* Теория тестирования выступает против необоснованного уровня доверия к серии успешно пройденных тестов. К сожалению, большинство установленных результатов теории тестирования – негативны, означая, по словам Дейкстры (Dijkstra), то, что “тестирование программы может использоваться для демонстрации наличия дефектов, но никогда не покажет их отсутствие”. Основная причина этого в том, что полное (всеобъемлющее) тестирование недостижимо для реального программного обеспечения.

1.2.6 Проблема неосуществимых путей (The problem of infeasible paths)

* Проблема автоматизированного тестирования связана с тем, что путь, по которому выполняются потоки работ тестируемой программной системы, не могут быть заданы входными параметрами.

1.2.7 Тестируемость (Testability)

* Степень легкости описания критериев покрытия тестами для заданной программной системы
* Возможность вероятность, возможность статистического измерения того, что при тестировании проявится сбой программной системы

1.3 Связь тестирования с другой деятельностью (Relationships of testing with other activities)

* Управление качеством
* Конструирование

2. Уровни тестирования (Test Levels)

2.1 Над чем производятся тесты (The target of the test)

* **Уровень тестирования** определяет “над чем” производятся тесты: над отдельным **модулем, группой модулей** или **системой,** в целом.

2.1.1 Модульное тестирование (Unit testing)

* Позволяет проверить функционирование отдельно взятого элемента системы.
* *Модуль* системы определяется контекстом.
* **IEEE 1008-87 “Standard for Software Unit Testing”**, (Интегрированная концепция систематического и документированного подхода к модульному тестированию)

2.1.2 Интеграционное тестирование (Integration testing)

* Уровень тестирования является процессом проверки взаимодействия между программными компонентами/модулями
* (стратегии в классике: “сверху-вниз”, “снизу-вверх”)

2.1.3 Системное тестирование (System testing)

* безопасность, производительность, точность, надежность
* тестируются интерфейсы к внешним приложениям, аппаратному обеспечению, операционной среде

2.2 Цели тестирования (Objectivies of Testing)

* Тестовые сценарии могут разрабатываться
  для проверки функциональных требований
  (функциональные тесты),
  для оценки нефункциональных требований
* Существуют такие тесты, когда количественные параметры и результаты тестов могут лишь опосредованно говорить об удовлетворении целям тестирования (“usability” – легкость, простота использования)

2.2.1 Приёмочное тестирование (Acceptance/qualification testing)

* Проверяет поведение системы на предмет удовлетворения требований заказчика.
* Это возможно в том случае, если заказчик берет на себя ответственность, связанную с проведением таких работ, как сторона “принимающая” программную систему, или специфицированы типовые задачи, успешная проверка (тестирование) которых позволяет говорить об удовлетворении требований заказчика.

2.2.2 Установочное тестирование (Installation testing)

* Данные тесты проводятся с целью проверки процедуры инсталляции системы в целевом окружении.

2.2.3 Альфа- и бета-тестирование (Alpha and beta testing)

* **альфа** (внутреннее пробное использование)
* **бета** (пробное использование с привлечением отобранных внешних пользователей)
* Отчеты об ошибках, поступающие от пользователей этих версий продукта, обрабатываются в соответствии с определенными процедурами, включающими подтверждающие тесты (любого уровня), проводимые специалистами группы разработки.

2.2.4 Функциональные тесты/тесты соответствия (Conformance testing/Functional testing/Correctness testing)

* Проверка соответствия системы, предъявляемым к ней требованиям, описанным на уровне спецификации поведенческих характеристик

2.2.5 Достижение и оценка надежности (Reliability achievement and evaluation)

* Помогая идентифицировать причины сбоев, тестирование подразумевает и **повышение надежности** программных систем.
* Случайно генерируемые сценарии тестирования могут применяться для **статистической оценки надежности.**

2.2.6 Регрессионное тестирование (Regression testing)

* Определение успешности регрессионных тестов (**IEEE 610-90 “Standard Glossary of Software Engineering Terminology”**) : “повторное выборочное тестирование системы или компонент для проверки сделанных модификаций не должно приводить к непредусмотренным эффектам”.

* Проблема регрессионного тестирования - **поиск компромисса между имеющимися ресурсами и необходимостью проведения таких тестов по мере внесения каждого изменения.**
* задача состоит в том, чтобы определить критерии “масштабов” изменений, с достижением которых необходимо проводить регрессионные тесты.

2.2.7 Тестирование производительности (Performance testing)

* Специализированные тесты проверки удовлетворения специфических требований, предъявляемых к параметрам производительности.
* Особый подвид тестов, когда делается попытка достижения количественных пределов, обусловленных характеристиками самой системы и ее операционного окружения.

2.2.8 Нагрузочное тестирование (Stress testing)

* Тестированием производительности с целью достижения ее реальных (достижимых) *возможностей* производительности и выполнением программной системы с повышением нагрузки, вплоть до достижения *запланированных характеристик* и далее, с отслеживанием поведения на всем протяжении повышения загрузки системы.

2.2.9 Сравнительное тестирование (Back-to-back testing)

* Единичный набор тестов, позволяющих сравнить две версии системы.

2.2.10 Восстановительные тесты (Recovery testing)

* Цель – проверка возможностей рестарта системы в случае непредусмотренной катастрофы (disaster), влияющей на функционирование операционной среды, в которой выполняется система.

2.2.11 Конфигурационное тестирование (Configuration testing)

* Если программное обеспечение создается для использования различными пользователями (в терминах “ролей”), данный вид тестирования направлен на проверку поведения и работоспособности системы в различных конфигурациях.

2.2.12 Тестирование удобства и простоты использования (Usability testing)

* Цель – проверить, насколько легко конечный пользователь системы может ее освоить, включая не только функциональную составляющую – саму систему, но и ее документацию;
* насколько эффективно пользователь может выполнять задачи, автоматизация которых осуществляется с использованием данной системы;
* насколько хорошо система застрахована (с точки зрения потенциальных сбоев) от ошибок пользователя.

2.2.13 Разработка, управляемая тестированием (Test-driven development)

* это не столько техника тестирования, сколько стиль организации процесса разработки, жизненного цикла, когда тесты являются неотъемлемой частью требований (и соответствующих спецификаций) вместо того, чтобы рассматриваться независимой деятельностью по проверке удовлетворения требований программной системой.
* TDD методология

* FDD – Feature-Driven Development (разработка на основе функциональных возможностей).
* TDD может естественно рассматриваться как составная часть XP или, как минимум Agile-методов.
* FDD может рассматриваться как один из методов гибкой разработки.

* *Тесты – инструмент* достижения характеристик системы, удовлетворяющей заданным требованиям, то есть потребностям пользователей, а “*возможности*” (features) – практически сами (чаще – функциональные) *требования*, воплощенные (в идеальном случае) в код.

3. Техники тестирования (Test Techniques)

* **3.1 Техники, базирующиеся на интуиции и опыте инженера (Based on the software engineer's intuition and experience)**

3.1.1 Специализированное тестирование (Ad hoc testing)

* Тесты основываются на опыте, интуиции и знаниях инженера, рассматривающего проблему с точки зрения имевшихся ранее аналогий. Данный вид тестирования может быть полезен для идентификации тех тестов, которые не охватываются более формализованными техниками.

3.1.2 Исследовательское тестирование (Exploratory testing)

* Одновременное обучение, проектирование теста и его исполнение.
* Заранее не определяется в плане тестирования и такие тесты создаются, выполняются и модифицируются динамически, по мере необходимости.
* Эффективность зависит от знаний инженера, формируемых на основе поведения тестируемого продукта в процессе проведения тестирования, степени знакомства с приложением, платформой, типами возможных сбоев и дефектов, рисками, ассоциированными с конкретным продуктом

3.2 Техники, базирующиеся на спецификации (Specification-based techniques)

3.2.1 Эквивалентное разделение <приложения> (Equivalence partitioning)

* Рассматриваемая область приложения разделяется на коллекцию наборов или эквивалентных классов, которые считаются эквивалентными с точки зрения рассматриваемых связей и характеристик <спецификации>. Репрезентативный набор тестов (только один тест) формируется из тестов эквивалентных классов (наборов классов).

3.2.2 Анализ граничных значений (Boundary-value analysis)

* Тесты строятся с ориентацией на использование тех величин, которые определяют предельные характеристики тестируемой системы. Расширением этой техники являются *тесты оценки живучести (robustness testing)* системы, проводимые с величинами, выходящими за рамки специфицированных пределов значений.

3.2.3 Таблицы принятия решений (Decision table)

* Таблицы представляют логические связи между условиями (могут рассматриваться в качестве “входов”) и действиями (могут рассматриваться как “выходы”).
* Набор тестов строится последовательным рассмотрением всех возможных кросс-связей в такой таблице.

3.2.4 Тесты на основе конечного автомата (Finite-state machine-based)

* Комбинация тестов для всех состояний и переходов между состояниями, представленных в соответствующей модели (переходов и состояний приложения).

3.2.5 Тестирование на основе формальной спецификации (Testing from formal specification)

* Для спецификации, определенных с использованием формального языка, возможно автоматически создавать и тесты для функциональных требований. Могут строится на основе модели, являющейся частью спецификации, не использующей формального языка описания.

3.2.6 Случайное тестирование (Random testing)

* Тесты генерируются случайным образом по списку заданного набора специфицированных характеристик.

3.3 Техники, ориентированные на код (Code-based techniques

3.3.1 Тесты, базирующиеся на блок-схеме (Control-flow-based criteria)

* Набор тестов строится исходя из покрытия всех условий и решений блок-схемы. В какой-то степени напоминает тесты на основе конечного автомата. Отличие – в источнике набора тестов.
* Максимальная отдача от тестов на основе блок-схемы получается когда тесты покрывают различные пути блок-схемы, сценарии потоков работ (поведения) тестируемой системы.
* Адекватность таких тестов оценивается как процент покрытия всех возможных путей блок-схемы.

3.3.2 Тесты на основе потоков данных (Data-flow-based criteria)

* Отслеживается полный жизненный цикл величин (переменных) – с момента рождения (определения), на всем протяжении использования, вплоть до уничтожения (неопределенности).
* В реальной практике используются нестрогое тестирование такого вида, ориентированное, например, только на проверку задания начальных значений всех переменных или всех вхождений переменных в код, с точки зрения их использования.

3.3.3 Ссылочные модели для тестирования, ориентированного на код (Reference models for code-based testing – flowgraph, call graph)

* Является не столько техникой тестирования, сколько контролем структуры программы, представленной в виде дерева вызовов (например, sequence-диаграммы, определенной в нотации UML и построенной на основе анализа кода).

3.4 Тестирование, ориентированное на дефекты (Fault-based techniques)

* На уровне названия таких техник тестирования, они, действительно, ориентированы на ошибки. Точнее – на специфические категории ошибок.

3.4.1 Предположение ошибок (Error guessing)

* Направлены на обнаружение наиболее вероятных ошибок, предсказываемых, например, в результате анализа рисков.

3.4.2 Тестирование мутаций (Mutation testing)

* **Мутация** – небольшое изменение тестируемой программы, произошедшее за счет частных синтаксических изменений кода (в частности, рефакторинга). Соответствующие тесты запускаются для оригинального и всех “мутировавших” вариантов тестируемой программы.
* SWEBOK фокусируется на возможности, с помощью тестов, определять отличия между мутантами и исходным вариантом кода. Если такое отличие установлено, мутанта “убивают”, а тест считается успешным. Обычно, данный подход фокусируется на синтаксических ошибках, на практике отслеживаемых современными средами разработки и, конечно, компиляторами.

3.5 Техники, базирующиеся на условиях использования (Usage-based techniques)

3.5.1 Операционный профиль (Operational profile)

* Базируется на условиях использования системы.
* Тестирование для оценки надёжности системы должно проводиться в таком тестовом окружении, которое максимально приближено к реальным условиям работы системы.
* Результаты таких тестов позволяют оценить поведение системы в реальных условиях.
* Входные параметры тестов задаются на основе вероятностного распределения соответствующих параметров или их наборов при эксплуатации (входные данные могут прогнозироваться исходя из частоты возможных сценариев работы пользователей).

3.5.2 Тестирование, базирующееся на надежности инженерного процесса (Software Reliability Engineered Testing)

* Базируется на условиях разработки системы.
* Соответствующие тесты (обозначаемые также аббревиатурой SRET от Software Reliability Engineered Testing)
* Проектируются в контексте используемого процесса разработки и методик тестирования.

3.6 Техники, базирующиеся на природе приложения (Techniques based on the nature of the application

* Объектно-ориентированное тестирование
* Компонентно-ориентированное тестирование
* Web-ориентированное тестирование
* Тестирование на соответствие протоколам
* Тестирование систем реального времени

3.7 Выбор и комбинация различных техник (Selecting and combining techniques)

3.7.1 Функциональное и структурное (Functional and structural)

* Техники тестирования, строящиеся на основе спецификаций или кода часто называют функциональными или структурными, соответственно. Оба подхода не должны противопоставляться, но дополнять друг друга.

3.7.1 Определенное или случайное (Deterministic vs. random)

* Тесты можно распределить по данным группам на основе используемой политики выбора или определения входных параметров тестов.

4. Измерение результатов тестирования (Test-related measures)

* Измерения являются инструментом анализа качества.
* Измерение результатов тестирования касается оценки качества получаемого продукта – программной системы.
* История измерений демонстрирует прогресс достижения приемлемого качества. Такая история является инструментом менеджмента качества.

4.1 Оценка программ в процессе тестирования (Evaluation of the program under test, IEEE 982.1-98)

4.1.1 Измерения программ как часть планирования и разработки тестов (Program measurements to aid in planning and design testing)

* Измерения могут базироваться на размере программ (например, в терминах количества строк кода или функциональных точек) или их структуре (например, с точки зрения оценки ее сложности в тех или иных архитектурных терминах). Структурные измерения могут также включать частоту обращений одних модулей программы к другим.

4.1.2 Типы дефектов, их классификация и статистика возникновения (Fault types, classification and statistics)

* Эффективность тестирования может быть достигнута в случае, понимания типов дефектов найденных в процессе тестирования программной системы и как изменяется их частота во времени (подразумевая историческую перспективу развития системы, а не её сбоев в процессе работы).
* Эта информация позволяет прогнозировать качество системы и помогает совершенствовать процесс разработки, в целом.

Стандарт IEEE 1044-93

* классифицирует возможные программные “аномалии”.

4.1.3 Плотность дефектов (Fault density)

* Тестируемая программа может оцениваться на основе подсчета и классификации найденных дефектов. Для каждого класса дефектов можно определить отношение между количеством соответствующих дефектов и размером программы (в терминах выбранных метрик оценки размера).

4.1.4 Жизненный цикл тестов, оценка надежности (Life test, reliability evaluation)

* Статистические ожидания в отношении надежности программной системы (см. выше 2.2.5 “Достижение и оценка надежности ”) могут использоваться для принятия решения о продолжении или прекращении (окончании) тестирования, исходя из заданных параметров приемлемого качества (например, плотности дефектов заданного класса).

4.1.5 Модели роста надежности (Reliability growth models)

* Модели обеспечивают возможности прогнозирования надежности системы, базируясь на обнаруженных сбоях (2.2.5). Модели такого рода разбиваются на две группы – *по количеству сбоев (failure-count)* и *времени между сбоями (time-between-failure)*.

4.2 Оценка выполненных тестов (Evaluation of the tests performed)

4.2.1 Метрики покрытия/глубины тестирования (Coverage/thoroughness measures)

* Критерии “адекватности” тестирования, в ряде случаев, требуют систематического выполнения тестов для определенных набора элементов программы, задаваемых ее архитектурой или спецификацией.
* Метрики позволяют оценить степень охвата характеристик системы (процент различных тестируемых параметров производительности) и глубину их детализации (случайное тестирование параметров производительности или с учетом граничных значений).
* Метрики помогают прогнозировать вероятностное достижение заданных параметров качества системы.

4.2.2 Введение искусственных дефектов (Fault seeding)

* Подход помогает классифицировать возможные ошибки и следующие за ними сбои, применяя в дальнейшем полученные результаты для моделирования (пусть, часто, и интуитивного) возможных причин реальных сбоев, обнаруженных в процессе тестирования.

4.2.3 Оценка мутаций (Mutation score)

* Получаемое в процессе тестирования мутаций (3.4.2) отношение “убитых” к общему числу сгенерированных мутантов помогает измерить эффективность выполняемых тестов.
* Количественные оценки мутаций имеют практическое значение только для определенных типов систем.

4.2.4 Сравнение и относительная эффективность различных техник тестирования (Comparison and relative effectiveness of different techniques)

* Возможные варианты интерпретации этого понятия –
* число тестов (данной техники), необходимых для обнаружения первого дефекта;
* отношение количества всех обнаруженных дефектов к дефектам, найденным с применением заданного подхода.

5. Процесс тестирования (Test Process)

* Концепции, стратегии, техники и измерения тестирования должны быть объединены в **единый процесс тестирования** как деятельности по обеспечению качества. Процесс тестирования поддерживает работы по тестированию и определяет “правила игры” для членов команды тестирования – от планирования тестов до оценки их результатов.

5.1 Практические соображения (Practical considerations)

5.1.1 Программирование без персоналий (Attitudes/Egoless programming)

* *Совместное стремление участников проекта обеспечить необходимое качество продукта*. Менеджеры играют ключевую роль в организации этой деятельности и на стадии разработки и в процессе сопровождения программных систем.

5.1.2 Руководства по тестированию (Test guides)

* Работы по тестированию могут руководствоваться различными соображениями и критериями – от управления рисками до специфицированных сценариев работы программных систем. В любом случае, желательно, исходя из ресурсов, количественных оценок и других характеристик, обеспечить использование различных техник тестирования для многосторонней оценки и улучшения качества получаемого продукта.

5.1.3 Управление процессом тестирования (Test process management)

* Работы по тестированию должны быть организованы в единый (однозначно интерпретируемый) процесс, на основе учета 4 элементов и связанных с ними факторов:
* людей (в том числе, в контексте организационной структуры и культуры),
* инструментов,
* регламентов
* количественных оценок (измерений).
* IEEE, ISO/IEC, ГОСТ Р 12207 не выделяет деятельность по тестированию в качестве самостоятельного процесса, однако, рассматривает соответствующие принципы работ по тестированию как неотъемлемую часть процессов жизненного цикла и сопровождения программных систем.
* IEEE 1074 деятельность по тестированию также объединена с другими оценочными работами как интегральная часть полного жизненного цикла.

5.1.4 Документирование тестов и рабочего продукта (Test documentation and work products)

* IEEE 829-98 “Standard for Software Test Documentation”:
* План тестирования
* Спецификация процедуры тестирования
* Спецификация тестов
* Лог тестов

5.1.5 Внутренние и независимые команды тестирования (Internal vs. independent test team)

* Формализация процесса тестирования может включать и организационную формализацию команд(ы) тестирования
* члены проектной команды, разрабатывающие код,
* внешние лица и группы.

5.1.6 Оценка стоимости и усилий, а также другие измерения процесса (Cost/effort estimation and other process measures)

* Ряд метрик, связанных с оценкой ресурсов, необходимых для тестирования, как и оценка эффективности тестирования на разных этапах и уровнях, основывается на точке зрения и практиках менеджмента проекта (подразделения, компании...) и используется для оценки и улучшения (оптимизации) процесса тестирования. Разные техники, концепции и модели тестирования требуют разных затрат – по времени и необходимым ресурсам. Результат – стоимость тестирования, как затратная составляющая проекта. Понимание соответствия между стоимостью/усилиями, необходимыми для той или иной формы тестирования является обязательной частью современного управления проектами разработки программного обеспечения.

5.1.7 Окончание тестирования (Termination)

* Тщательные измерения, такие как достигнутое покрытие кода тестами или охват функциональности, безусловно, очень полезны. Однако, сами по себе они не могут определить критериев достаточности тестирования. Приятие решения об окончании тестирования также включает рассмотрение стоимости и рисков, связанных с потенциальными сбоями и нарушениями надёжности функционирования тестируемой программной системы. В то же время, стоимость самого тестирования также является одним из ограничений, на основе которых принимается решение о продолжении тех или иных связанных с проектом работ (с частности, тестирования) или об их прекращении. (1.2.1).

5.1.8 Повторное использование и шаблоны тестов (Test reuse and test patterns)

* Доведение тестов до конца и обеспечение сопровождения программной системы необходимо каждый фрагмент системы тестировать систематическим образом, повторно используя наработанные тесты. Общий репозиторий тестовых активов должен находиться под контролем системы конфигурационного управления, с тем, чтобы любые изменения в требованиях или дизайне могли быть отражены в используемых наборах тестов, в том числе, с точки зрения их расширения новыми тестами, если этого требуют соответствующие изменения.
* Шаблоны тестов конструируются на основе тестовых решений, наработанных для проверки определенных ситуаций или типовых фрагментов программных систем. Такие шаблоны должны быть документированы с учетом повторного использования, включая прозрачные возможности их адаптации под специфику программных решений, к которым такие шаблоны применяются.

5.2 Тестовые работы (Test Activities)

* Успешное управление тестовыми работами зависит от процессов конфигурационного управления (Software Configuration Management)

5.2.1 Планирование (Planning)

* координацию персонала
* управление оборудованием и другими средствами, необходимыми для организации тестирования
* планирование обработки нежелательных результатов (т.е. является управлением определенными видами рисков)

5.2.2 Генерация сценариев тестирования (Test-case generation)

* Создание тестовых сценариев основывается на уровне и конкретных техниках тестирования. Тесты должны находиться под управлением системы конфигурационного управления и описывать ожидаемые результаты тестирования.

5.2.3 Разработка тестового окружения (Test environment development)

* Используемое для тестирования окружение должно быть совместимо с инструментами программной инженерии (будут рассматриваться позднее как тема самостоятельной области знаний). Это окружение должно обеспечивать разработку и контроль тестовых сценариев, ведение журнала тестирования, и возможности восстановления ожидаемых и отслеживаемых результатов тестирования, самих сценариев, а также других активов тестирования.

5.2.4 Выполнение тестов (Execution)

* должны фиксироваться все работы и результаты процесса тестирования
* форма журналирования таких работ и их результатов должна быть такой, чтобы соответствующее содержание было понятно, однозначно интрепретируемой и повторяемо другими лицами (не теми, кто первоначально проводил тестирование)
* тестирование должно проводиться в соответствии с заданными и документированными процедурами
* тестирование должно производиться над однозначно идентифицируемой версией и конфигурацией программной системы
* IEEE 1008-87

5.2.5 Анализ результатов тестирования (Test results evaluation)

* Для определения успешности тестов их результаты должны оцениваться, анализироваться. В большинстве случаев, “успешность” тестирования подразумевает, что тестируемое программное обеспечение функционирует так, как ожидалось и в процессе работы не приводит к непредусмотренным последствиям. Не все такие последствия обязательно являются сбоями, они могут восприниматься как “помехи”. Однако, любое непредусмотренное поведение может стать источником сбоев при изменении конфигурации или условий функционирования системы, поэтому требуют внимания, как минимум, с точки зрения идентификации причин таких помех. Перед устранением обнаруженного сбоя, необходимо определить и зафиксировать те усилия, которые необходимы для анализа проблемы, отладки и устранения. Это позволит в дельнейшем обеспечить большую глубину измерений, а, соответственно, в перспективе, иметь возможность улучшения самого процесса тестирования. В тех случаях, когда результаты тестирования особенно важны, например, в силу критичности обнаруженного сбоя, может быть сформирована специальная группа анализа (review board).

5.2.6 Отчёты о проблемах/журнал тестирования (Problem reporting/Test log)

* В процессе тестовой деятельности ведётся журнал тестирования, фиксирующий информацию о соответствующих работах: когда проводится тест, какой тест, кем проводится, для какой конфигурации программной системы (в терминах параметров и в терминах идентифицируемой версии контекста конфигурационного управления) и т.п. Неожиданные или некорректные результаты тестов могут записываться в специальной подсистеме ведения отчетности по сбоям (problem-reporting system, обеспечивая формирование базы данных, используемой для отладки, устранения проблем и дальнейшего тестирования. Кроме того, аномалии (помехи), которые нельзя идентифицировать как сбои, также могут фиксироваться в журнале и/или системе ведения отчетности по сбоям. В любом случае, документирование таких аномалий снижает риски процесса тестирования и помогает решать вопросы повышения надежности самой тестируемой системы. Отчёты по тестам могут являться входом для процесса управления изменениями и генерации запросов на изменения (change request) в рамках процессов конфигурационного управления (см. далее соответствующую область знаний “Software Configuration Management”).

5.2.7 Отслеживание дефектов (Defect tracking)

* Сбои, обнаруженные в процессе тестирования, чаще всего порождаются дефектами и ошибками, присутствующими в тестируемой программной системе (также они могут быть следствием поведения операционного и/или тестового окружения). Такие дефекты могут (и, чаще всего, должны) анализироваться для определения момента и места первого появления данного дефекта в системе, какие типы ошибок стали причиной этих дефектов (например, плохо сформулированные требования, некорректный дизайн, утечки памяти и т.д.) и когда они могли бы быть обнаружены впервые. Вся эта информация используется для определения того, как может быть улучшен сам процесс тестирования и насколько критична необходимость таких улучшений.