**Тестирование программного обеспечения** — процесс исследования, испытания программного продукта, имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации

### Виды тестирования

Все **виды тестирования программного обеспечения**, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

* Функциональные
* Нефункциональные
* Связанные с изменениями

#### Функциональные виды тестирования

Функциональные тесты базируются на функциях и особенностях, а также взаимодействии с другими системами, и могут быть представлены на всех уровнях тестирования: компонентном или модульном (Component/Unit testing), интеграционном (Integration testing), системном (System testing) и приемочном (Acceptance testing). Функциональные виды тестирования рассматривают внешнее поведение системы. Далее перечислены одни из самых распространенных видов функциональных тестов:

* Функциональное тестирование (Functional testing)
* Тестирование безопасности (Security and Access Control Testing)
* Тестирование взаимодействия (Interoperability Testing)

#### Нефункциональные виды тестирования

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, "Как" система работает. Далее перечислены основные виды нефункциональных тестов:

* Все виды тестирования производительности:
  + нагрузочное тестирование (Performance and Load Testing)
  + стрессовое тестирование (Stress Testing)
  + тестирование стабильности или надежности (Stability / Reliability Testing)
  + объемное тестирование (Volume Testing)
* Тестирование установки (Installation testing)
* Тестирование удобства пользования (Usability Testing)
* Тестирование на отказ и восстановление (Failover and Recovery Testing)
* Конфигурационное тестирование (Configuration Testing)

#### Связанные с изменениями виды тестирования

После проведения необходимых изменений, таких как исправление бага/дефекта, программное обеспечение должно быть пере тестировано для подтверждения того факта, что проблема была действительно решена. Ниже перечислены виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществленного исправления дефекта:

* Дымовое тестирование (Smoke Testing)
* Регрессионное тестирование (Regression Testing)
* Тестирование сборки (Build Verification Test)
* Санитарное тестирование или проверка согласованности/исправности (Sanity Testing)

**Понятие дымовое тестирование** пошло из инженерной среды:

"При вводе в эксплуатацию нового оборудования ("железа") считалось, что тестирование прошло удачно, если из установки не пошел дым."

В области же программного обеспечения, *дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции*.

**Тестирование сборки:**

Тестирование, направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования. По своим целям является аналогом Дымового Тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.

Тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения. Уровень тестирования определяет то, **над чем** производятся тесты: над отдельным модулем, группой модулей или системой, в целом. Проведение тестирования на всех уровнях системы - это залог успешной реализации и сдачи проекта.

### Уровни Тестирования

1. Компонентное или Модульное тестирование (Component Testing or Unit Testing)
2. Интеграционное тестирование (Integration Testing)
3. Системное тестирование (System Testing)
4. Приемочное тестирование (Acceptance Testing)

*Компонентное (модульное) тестирование* проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.). Обычно компонентное (модульное) тестирование проводится, вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки. Все найденные дефекты, как правило исправляются в коде без формального их описания в системе менеджмента багов (Bug Tracking System).

Один из наиболее эффективных подходов к компонентному (модульному) тестированию - это *подготовка автоматизированных тестов* до начала основного кодирования (разработки) программного обеспечения. Это называется разработка от тестирования (test-driven development) или подход тестирования вначале (test first approach). При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор, пока все тесты не будут успешно пройдены.

### Разница между компонентным и модульным тестированием

По-существу эти уровни тестирования представляют одно и тоже, разница лишь в том, что в компонентном тестировании в качестве параметров функций используют реальные объекты и драйверы, а в модульном тестировании - конкретные значения.

**Интеграционное тестирование** предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

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. Для минимизации рисков, связанных с особенностями поведения в системы в той или иной среде, во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи.

Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

* определения удовлетворяет ли система приемочным критериям;
* вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет.

### Тест дизайн

**Тест дизайн** – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

#### План работы над тест дизайном

* анализ имеющихся проектных артефактов: документация (спецификации, требования, планы), модели, исполняемый код и т.д.
* написание спецификации по тест дизайну (Test Design Specification)
* проектирование и создание тестовых случаев (Test Cases)

#### Роли, ответственные за тест дизайн

* Тест аналитик - определяет "**ЧТО тестировать?**"
* Тест дизайнер - определяет "**КАК тестировать?**"

Попросту говоря, задача тест аналитиков и дизайнеров сводится к тому, чтобы, используя различные стратегии и техники тест дизайна, создать набор тестовых случаев, обеспечивающий оптимальное тестовое покрытие тестируемого приложения. Однако, на большинстве проектов эти роли не выделяется, а доверяется обычным тестировщикам, что не всегда положительно сказывается на качестве тестов, тестировании и, как из этого следует, на качестве программного обеспечения (конечного продукта).