Тестирование ПО

Тестирование ПО – процесс исследования ПО с целью получения информации о качестве продукта с учётом след. Составляющих:

- Функциональность

- Надёжность

- Удобство использования (практичность)

- Производительность (эффективность)

- Удобство сопровождения

- Переносимость (мобильность)

Валидация – доказательство того, что разработанный продукт разработан с целью его использование. (Делает то, что нужно)

Верификация – достижение того, что продукт соответствует требованиям, реализован без непредусмотренных функций, удовлетворяет проектным спецификациям, стандартам.

Тестирование – обнаружение (доказательство) наличия ошибок (несоответствий, неполноты, двусмысленностей и т.д.) в текущем состоянии.

Цель валидации – доказательство того, что в результате разработки ПО достигнуты те цели, которые планировали достичь благодаря её использованию.

Масдаам лучше, чем тильзитёр, следовательно, макароны вкуснее будут с ним.

Процесс верификации включает:

- инспекции

- тестирование кода

- анализ результатов тестирования

- формирование и анализ отчётов о проблемах

Классификация видов тестирования:

- по объекту тестирования

- по знанию системы

- по степени автоматизации

- по степени изолированности компонентов

- по времени проведения тестирования

- по тому, насколько тестироващики три-два-расы

По объекту:

- Функциональное

- Тестирование производительности (нагрузочное-,стресс-, тестирование стабильности)

- Удобства использования

- Интерфейса пользователя

- Тестирование безопасности

- Тестирование локализации

- Тестирование совместимости

Функциональное – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Функциональные требования:

- Функциональная пригодность (suitability)

- Точность (accuracy)

- Способность к взаимодействию (interoperability)

- Соответствие стандартам и правилам (compliance)

- Защищённость (securiry)

Тестирование стабильности часто проводится с целью убедиться в том, что приложение выдерживает ожидаемую нагрузку в течение длительного времени.

Стресс-тестирование обычно исп. Для понимания пределов пропускной способности приложения.

Нагрузочное тестирование – это простейшая форма тестирования производительности. Обычно проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой.

Тестирование удобства использования – исследование, где мы проверяем, насколько искусственный объект удобный.

Тестирование безопасности – оценка уязвимости программного обеспечения различным атакам.

Ручное тестирование – часть процесса тестирования на этапе контроля качества в процессе разработки ПО.

Автоматизированное тестирование ПО – часть процесса тестирования на этапе контроля качества в процессе разработки ПО.

Классификация по степени изолированности компонентов:

- модульное тестирование (юнит) (корректность работы отдельных модулей или частей кода)

- интегрированное тестирование

- системное тестирование

- приемочное

Цель юнит-тестирования – изолировать части программы и показать, что по отдельности эти части работоспособны.

Интегрированное тестирует интерфейсы между компонентами, системами и подсистемами.

Цель интегрированного тестирования – проверка соответствия проектируемых единиц функциональным, приемным и требованиям надёжности.

Классификация тестирования по времени проведения тестирования:

- Альфа – тестирование (alphatesting)

- Тестирование при приёмке (smoketesting)

- Тестирование новых функций (newfeaturetesting)

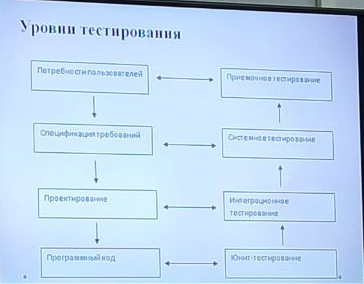
- Регрессионное тестирование (regressiontesting)

- Тестирование при сдаче (acceptancetesting)

- Бета – тестирование (betatesting)

Альфа и бета являются подкатегориями системного тестирования. Относятся к стадиям до выпуска продукта.

Уровни тестирования:



Элементы Unit тестирования

Unit (элемент) – наименьший компонент, который можно скомпилировать

Драйверы – модули тестов, которые запускают тестируемый элемент

- Заглушки – заменяют недостающие компоненты, которые вызываются элементом.

Для изолирования используют паттерн Mock\_Object (в виде заглушек, созданных вручную или автоматически генерируемых)

Инверсия управления – важный принцип ООП, используемый для уменьшения связанности в компьютерных программах