13 Верификация программного обеспечения - методы инспектирования, тестирования и отладки

верификация и аттестация - методы проверки и анализа, в ходе которых проверяется соответствие результатов разработки требованиям заказчика.

верификация - правильно ли построено программное средство (соответствие спецификации)
аттестация - правильно ли работает система (соответствие ожиданиям заказчика)

методики нахождения несоответствий:

- инспектирование
- тестирование

после нахождения - отладка (исправление несоответствий)

инспектирование

статический подход, проверка без выполнения самой программы.
объекты: документы, модели, программный код.
выполнение программы как правильно дорогостоящий процесс, инспектирование его удешевляет.
за сеанс инспектирования находится несколько дефектов, тестирования - 1 ошибка.

специализируется на конкретных методах: стиль кодирования, утечки памяти, соответствие спецификации.

инспектор тратит час на проверку, предоставляет список дефектов. можно найти более 60% ошибок. на собрании назначаются задачи для отладки и исполнители.
однако статический метод не может проверить правильность функционирования, производительность, информационную безопасность.

тестирование

динамический подход, запуск программы.
проверяет правильность внутренней организации вычислений (компонент ПО) и правильность работы на различных входных данных (проверка выходных данных).
результат - нахождение ошибок, устранение которых повысит качество ПО.

объекты тестирования - программный код, прототип.

1. выполнение программы
2. хороший тест с большой вероятностью находит ошибку
3. тестирование не может показать отсутствие ошибок (не является доказательством их отсутствия)

отладка

- локализация и устранение ошибок
- следствие тестирования/инспектирования
- журнал выполнения тестирования

повторное тестирование может обнаружить ошибку, поскольку ПО постоянно меняется и дорабатывается.

методы тестирования

- метод черного ящика - знаем, что ПО должно делать, демонстрируем на соответствующих тестах - проверка требований

- метод белого ящика - знаем, как ПО устроено изнутри, проверяем правильность работы - проверка проектных решений

план тестирования создается задолго до самого тестирования, на этапе проектирования.
число тестов бесконечно, всеобъемлющее тестирование невозможно.

разработчики и тестировщики должны работать отдельно.

метод белого ящика

1. тестирование ветвей путем выполнения программы
   потоковый граф
   цель: разработать такой набор тестов, в котором любой оператор выполняется минимум 1 раз, то есть пути покрывают все вершины (операторы) графа.
2. тестирование циклов
   цель: разработать такой набор тестов, чтобы покрыть все варианты выполнения цикла.
   пропуск цикла, однократное выполнение, выполнение m раз (m - типовое значение, сильно отходящее от левой и правой границы). n - макс. кол-во итераций: m=n-1, n, n+1 (проверка граничных значений)
3. тестирование состояний
   тоже можно представить в виде графа
   цель: разработать такой набора тестов, чтобы пройти через каждое состояние хотя бы один раз.
4. тестирование условий if-then-else
5. тестирование потоков данных
6. тестирование на устойчивость к ошибкам

метод черного ящика

1. разбиение входных данных по классам (не пересекаются и в сумме полностью покрывают). для каждого класса не менее 1 теста.
2. анализ граничных значений - большая часть ошибок приходится на границы входных данных.
3. сравнительное тестирование (back to back) - используется для нескольких реализаций, результаты не совпали ⇒ в одной из реализаций есть ошибки.

тестирование никогда не заканчивается, просто переходит от разработчиков к пользователям. любой запуск программы это тест.