Error: Acción humana que produce un resultado incorrecto.

- Introduce un defecto en el software, que se manifiesta a través de fallas
- Es humano
- Puede venir del programador o del usuario que no usa el software de la manera correcta

Defecto: presencia de una imperfección que puede ocasionar fallas.

- Bug
- Esta en el código

Falla: comportamiento observable incorrecto con respecto a los requisitos

- Es un comportamiento no deseado

Verificación: ¿Estoy construyendo bien?
Validación: ¿Estoy construyendo el producto correcto?

Principios

Testing exhaustivo es imposible: No se pueden probar todos los casos, solo se testean los casos prioritarios
Agrupación de defectos: Un pequeño grupo de módulos contiene la mayoría de los defectos descubiertos
Paradoja del pesticida: Si se usan las mismas pruebas todo el tiempo se dejará de encontrar bugs, hay que ir actualizándolas (como las plantas resisten pesticida)
Testing muestra la presencia de defectos: Solo muestra la presencia de defectos, no la ausencia de estos
Falacia de ausencia de errores: No sirve encontrar errores si el software no sirve o es malo
Testing depende del contexto: como se aplica el testing depende del contexto, si es algo críticos se testeará más que si es solo comercial.
Parsing temprano: Se debe comenzar a testear lo antes posible

QA (Quality Assurance) > QC (Qualitiy Control) > Testing

Orden Testing:

1. Test Planning: Definir los objetivos a testear, alcance, riesgos, técnicas, política
2. Test Monitor and Control: Contrasta el resultado con el criterio especificado
3. Test Analysis: Se identifican features a evaluar (?)
4. Test Design: Se diseñan y priorizan los casos de prueba. Setup ambiente pruebas
5. Test Implementation: Creación de scripts automaticos de pruebas
6. Test Execution: Ejecución de los casos de pruebas
7. Test Completion: Revisar si todos los requirimientos se cumplen

Software Life Cycle: Requerimientos – Diseño – Construcción – Pruebas – Mantenimiento

Test Unitario o de Componente: Reduce el riesgo, verificar si componente funciona.

- Objetivo: Componentes, unidades, módulos, clases, EDD

Test de integración:

- Componente: Interacción entre componentes, después del component testing
- Sistema: Interacción entre destinos sistemas, después de sistema testing
- Objetivo: subsistemas, infraestructuras, implementación de BDD, APIs, microservicios
  - Tipos: **Big Bang** (todos los componentes en un solo paso), **Top-Down** (Siguiendo el flujo de la arquitectura, componentes generalmente sustituidos por stubs), **Bottom-Up** (Flujo de abajo hacia arriba, sustituyendo componentes por drivers)

Test de sistema: Prueba el sistema como un todo

- Objetivo: Aplicaciones, hardware/software, SSOO

Test de aceptación: Valida con respecto a las necesidades del usuario, requerimientos y procesos de negocio para ver si se acepta o no el sistema

  - Tipos: **Operational** (valida requerimientos para operación, aplicado por usuarios y adminstradores de la aplicación), **Contrato** (valida el criterio de un contrato, definición formar del aceptado), **Regulación** (valida regulaciones que debe seguir)
- Test Alpha: Usuarios lo prueban frente a los desarrolladores
- Test Beta: Usuarios lo prueban solos en un ambiente real

Tipos de testing:

- Functional testing: valuar calidad (completitud y correctitud)
- Non-functional testing: confiabilidad, eficiencia, compatibilidad y usabilidad
- White box testing: arquitectura/estructura
- Change related testing: efecto de los cambios
  - Confirmation tesing: confirmar que defectos fueron arreglados
  - Regression testing: buscar cambios inesperados