

Pruebas de Software y Casos de Prueba

Semana 13

Agenda

- Introducción
- Validación vs verificación
- Pruebas de Desarrollo
- Pruebas unitarias
- Selección de casos de pruebas

Introducción

Accidentes famosos

- **ARIANE 5 - 1996:** Cohete explotó 37 segundos después del lanzamiento.
- **Chernobyl – 1986:** Desastre nuclear durante la guerra fría.
- **Orbiter Climate Mars – 1999:** El satélite debía recolectar información de cambios climáticos y atmosféricos en Marte, pero se terminó estrellando.

¿Qué tienen en común estos desastres?

Fuente: <https://outfresh.com/knowledge-base/6-famous-software-disasters-due-lack-testing/>

Accidentes famosos

- **ARIANE 5 - 1996:** Se usó el sistema de control del Ariane 4, el cual funcionaba con menor precisión de números flotantes.
- **Chernobyl – 1986:** Desactivaron el Sistema de enfriamiento de emergencia mientras hacían pruebas con el nuevo reactor.
- **Orbiter Climate Mars – 1999:** El sistema de aterrizaje no trabajaba con las mismas unidades de medida estándar y se terminó estrellando.

Fuente: <https://outfresh.com/knowledge-base/6-famous-software-disasters-due-lack-testing/>



Remember the Mars Climate Orbiter incident from 1999?

Fuente: [Slideplayer.com](https://www.slideplayer.com)

Introducción

¿Las pruebas de software podrían ser de vida o muerte?

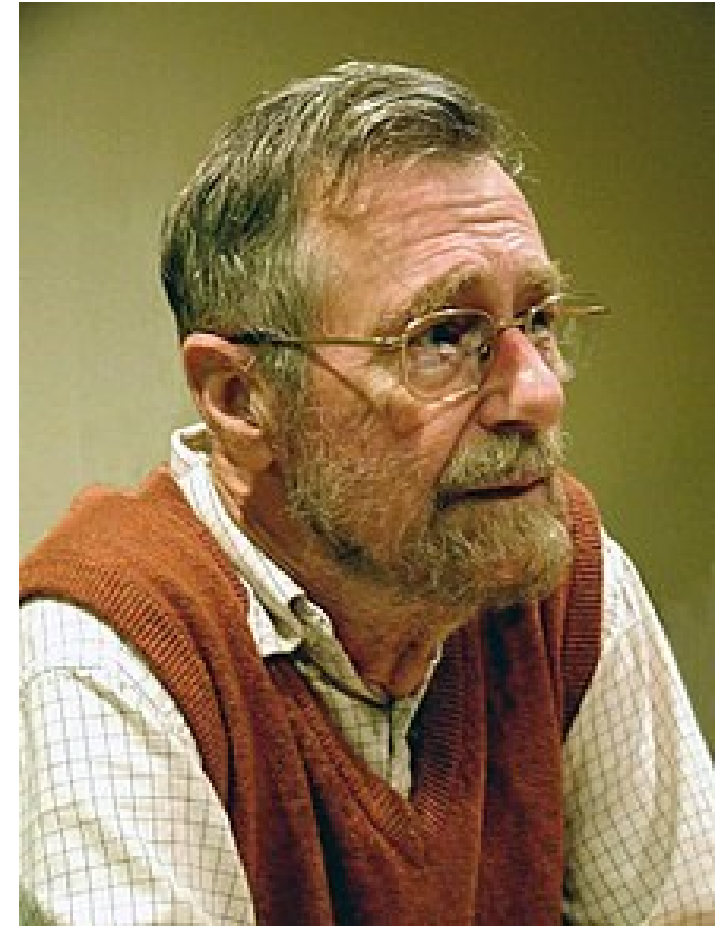
- Sí.

Es necesario realizar las pruebas para:

- Probar el correcto funcionamiento del software (Calidad)
- Encontrar anomalías en el flujo de datos (Encontrar problemas)
- Evitar problemas futuros

Introducción

“Testing can only show the presence of errors, not their absence. ... Everyone knows that testing can only show that errors exist, not that they don’t.”



Edsger W. Dijkstra, Ph.D.

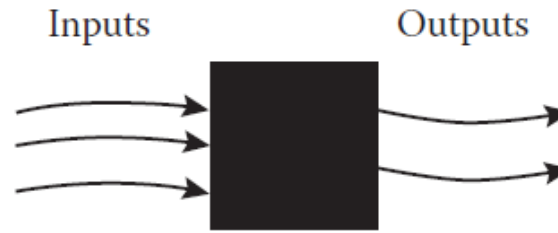
Definiciones (Elija el orden correcto, lista derecha)

-
- The diagram consists of six colored arrows connecting the terms on the left to their definitions on the right:
- A yellow arrow points from 'A. Error' to '1. Es una alerta al usuario de que existe una falla.'
 - A green arrow points from 'B. Fault' to '2. Bug. Se pueden propagar en las etapas de desarrollo.'
 - An orange arrow points from 'C. Failure' to '4. Una falla ocurre cuando se ejecuta un código defectuoso.'
 - A blue arrow points from 'D. Incident' to '3. Caso de prueba está asociado a un comportamiento del sistema.'
 - A grey arrow points from 'E. Test case' to '5. Defecto es el resultado de un error.'
 - A dark blue arrow points from 'F. Test' to '6. El proceso de ejecutar los casos de prueba.'
- A. Error
- B. Fault
- C. Failure
- D. Incident
- E. Test case
- F. Test
1. Es una alerta al usuario de que existe una falla.
2. Bug. Se pueden propagar en las etapas de desarrollo.
3. Caso de prueba está asociado a un comportamiento del sistema.
4. Una falla ocurre cuando se ejecuta un código defectuoso.
5. Defecto es el resultado de un error.
6. El proceso de ejecutar los casos de prueba.

Test Case

- Testing está compuesto principalmente de un conjunto de casos de prueba para el elemento que se desea probar.
- Características:
 - Identificador
 - Propósito
 - Precondiciones
 - Entradas y salidas esperadas
 - Postcondiciones
 - Historial de ejecuciones

Test cases

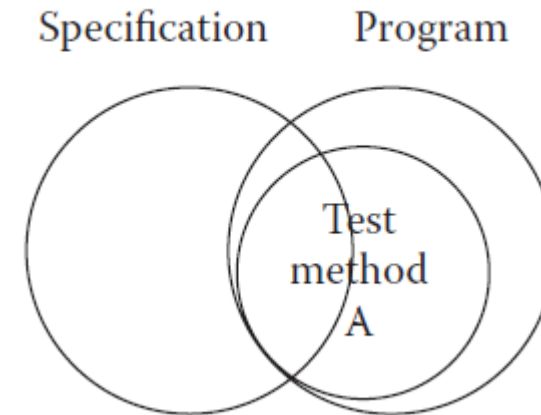
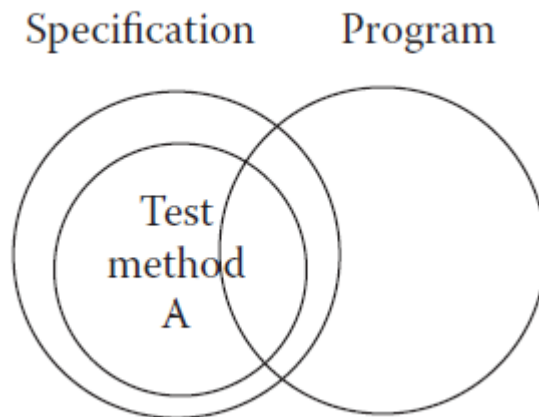


Specification-Based testing

- Entradas y salidas
- Black box / Código no se conoce
- Casos redundantes

Code-Based testing

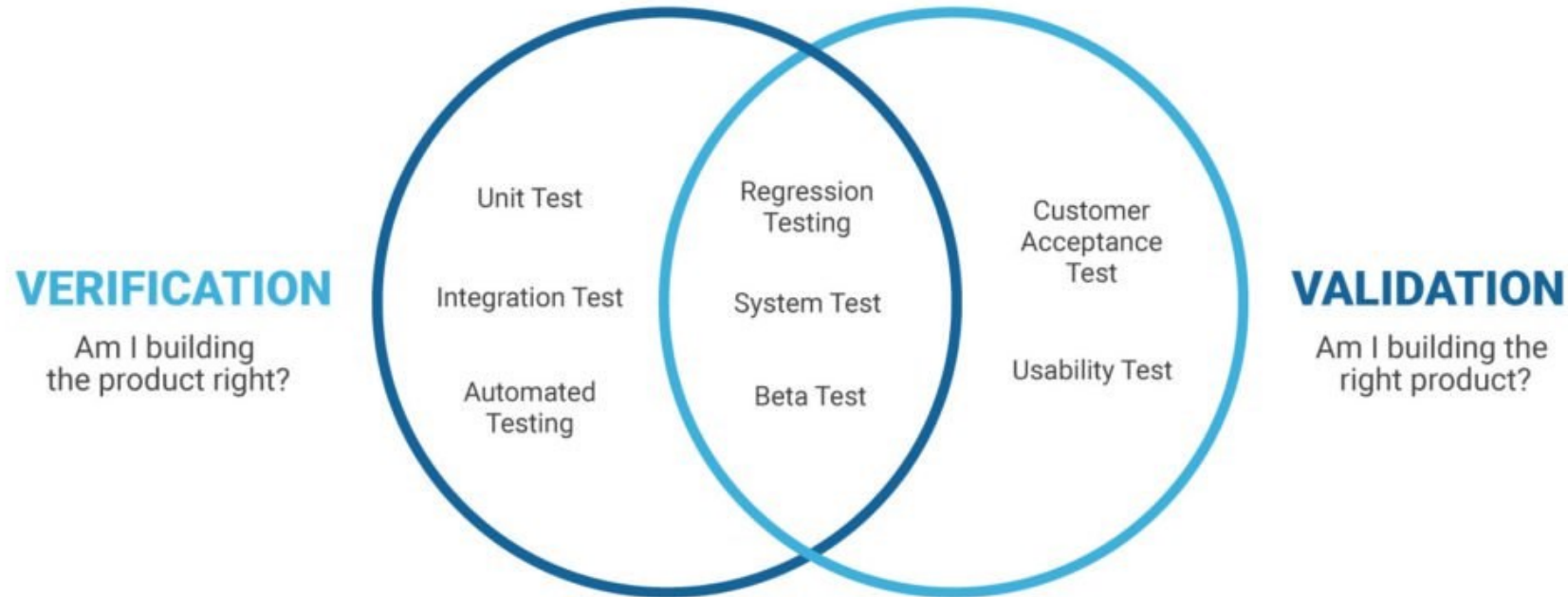
- Clear box
- Identifica los casos de prueba conociendo el código Fuente



Validación vs verificación

Validación vs verificación

- Software desarrollado tiene la menor cantidad de fallos, y
- Funciona tal y como se indica en los requerimientos



Fuente: <https://www.softwareautomationhelp.com>

Validación vs verificación

- El objetivo principal del proceso de validación y verificación es indicar que el **Software** es **suficientemente bueno** para ser utilizado en la vida real.
- El proceso (V&V) verifica que el software realiza lo que se supone debería realizar (según los requerimientos).
- Las pruebas del sistema son parte del proceso de (V & V).
- Las 3 principales etapas del proceso de pruebas son:
 - **Development testing** ← Nos enfocaremos en este
 - Release testing
 - User testing

Pruebas de Desarrollo

Pruebas de desarrollo

- Pruebas realizadas durante el desarrollo
- Realizadas por los diseñadores y programadores del Sistema
- Principalmente intenta identificar fallos y bugs.
- Comprende tres tipos:
 - **Unit testing** ← Solo analizaremos estos
 - Component testing
 - System testing

Pruebas de desarrollo

- Pruebas unitarias:
 - Probar unidades de software individualmente. (Métodos, rutinas u objetos)
 - Los test deben llamar a los métodos o funciones con diferentes parámetros de entradas.
 - Se deben diseñar los tests para cubrir todas las características de un objeto.
 - Normalmente son automatizadas (pero pueden ejecutarse manualmente)
 - Forman parte del mismo repositorio / paquete del código que verifica.

Pruebas unitarias

- Usualmente los métodos deben ser probados de forma aislada, sin embargo, en ciertos casos se debe probar en secuencia.
- Se debe tratar de que cada prueba unitaria sea:
 - Repetible
 - Independiente
 - Automático
- Se deben cubrir varios escenarios.

Pruebas unitarias

- Partes de una prueba:
 - Setup part. Entradas y salidas esperadas.
 - Call part. Llamar al método a ser probado.
 - Assertion part. Comparar resultado obtenido con salida esperada.

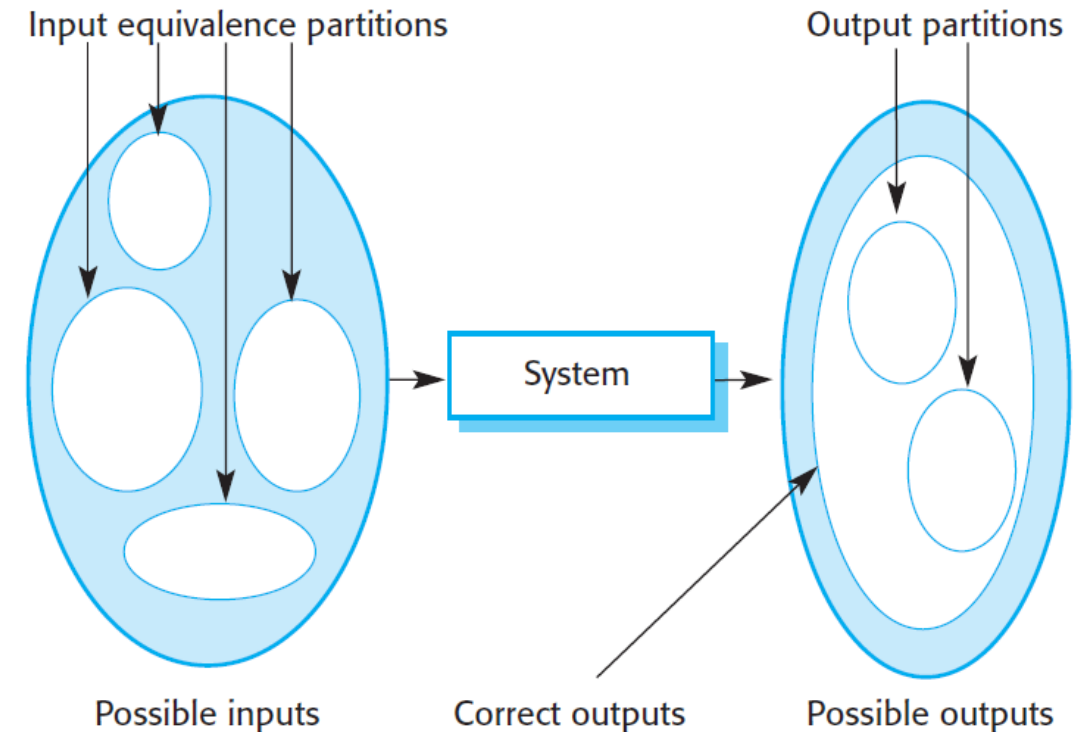
Selección de casos de pruebas

Selección de casos de pruebas

- Las pruebas pueden consumir mucho tiempo, por lo tanto deben seleccionarse los **casos de prueba efectivos**.
- Tipos de casos de pruebas:
 - Deben probar que cuando se usa un método en un entorno normal, funciona como es debido. Muestra el **funcionamiento normal** del programa.
 - Deben mostrar cuando existen fallos o defectos en una rutina o método. Generalmente están basados en la experiencia del programador de la prueba, sobre los **errores más comunes** que se podrían presentar.

Selección de casos de pruebas

- Se deben buscar grupos de datos **equivalentes** (óvalos blancos).
- Se espera que el sistema se **comporte igual** para los datos que pertenecen a una mismo grupo.
- Se debe escoger los **datos de los límites** de cada grupo/partición equivalente.



Selección de casos de pruebas

Some of the most general guidelines suggested are:

- Choose inputs that force the system to generate all error messages.
- Design inputs that cause input buffers to overflow.
- Repeat the same input or series of inputs numerous times.
- Force invalid outputs to be generated.
- Force computation results to be too large or too small.

Antes de finalizar

Puntos para recordar

- Las pruebas solo indican que no se han encontrado errores.
- Validación vs verificación
 - El proceso de pruebas es parte del proceso de V&V
- Pruebas de Desarrollo
 - Son principalmente para identificar fallos en la programación.
- Pruebas unitarias
 - Probar rutinas de forma individual, repetible y automatizada.
- Selección de casos de pruebas
 - Probar datos de los bordes y del medio de cada grupo equivalente.

Lectura adicional

- Ian Sommerville, “Software Engineering
 - Chapter 8: Software testing.
- Verification vs validation.
 - <https://www.softwareautomationhelp.com/difference-between-verification-and-validation/>

Próxima sesión

- Frameworks para pruebas unitarias.