Pruebas

José Galaviz

¿Qué es *testing*?

- Hacer pruebas a un programa o módulos de programación para detectar posibles errores.
- Se pretende con ello adquirir un cierto nivel de confianza en el software.
- Una cierta estimación de la calidad del software.

¿Qué no es *testing*?

- No garantiza la ausencia de errores, sólo prueba su existencia.
- No se pretenden corregir ni explicar los errores, sólo hacerlos evidentes.
- La prueba no mejora la calidad, sólo es un indicador de ella.

Tipos
(según su ámbito)

- Pruebas unitarias (*Unit testing*).
- Pruebas de componentes (*Component testing*).
- Pruebas de integración (*Integration testing*).
- Pruebas de regresión (*Regression testing*).
- Pruebas de sistema (*System testing*).

Unitarias

Para probar una única clase (POO), una rutina o rutinas pertenecientes al mismo módulo, hechas por un mismo programador o un equipo pequeño de programadores.

De componente

Para probar un conjunto de clases, paquete o rutinas relacionadas elaboradas por uno o varios equipos de programadores.

De integración

Para probar la correcta interacción entre diferentes clases o módulos, creados por múltiples equipos de programadores.

De regresión

Repetición de una o varias pruebas de los tipos previos tras una modificación.

De sistema

Pruebas globales con el sistema en su configuración final.

Tipos
(según su conocimiento del programa)

- De caja negra (*black box test*).
- De caja blanca o transparente (*white / glass box test*).

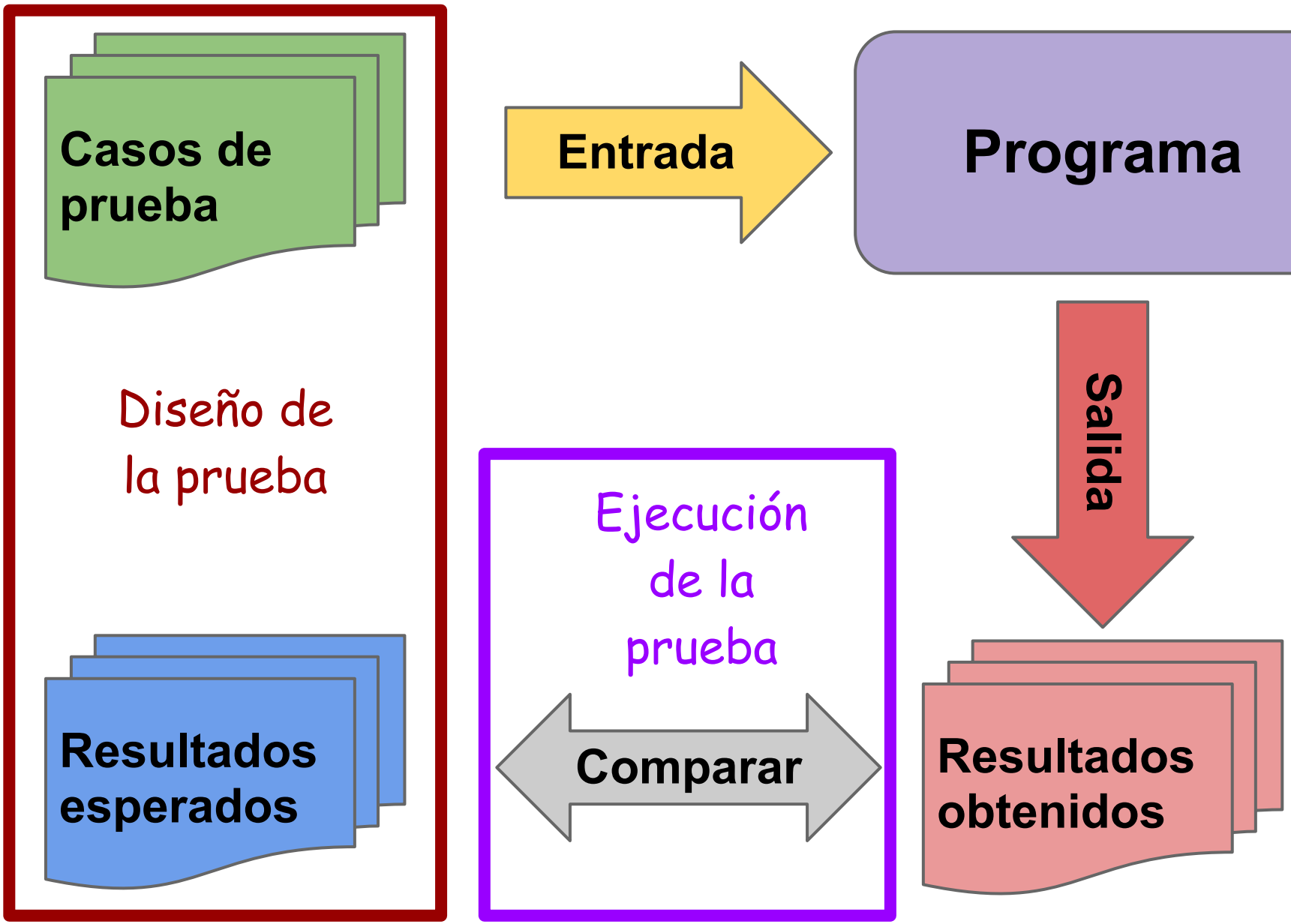
De caja negra

Sin conocer los detalles de implementación de la pieza de software que se probará. Típicamente cuando quien prueba no es quien desarrolló ni puede o debe ver el código fuente.

De caja transparente

Conociendo los detalles de implementación del software a probar. Típicamente cuando quien prueba es quien desarrolló o se tiene acceso al código fuente.

Esquema general



¿Cuando?

- Lo mejor es diseñar las pruebas unitarias antes o durante el proceso de elaboración del software que debe pasar las pruebas.

Consejos

- Escribir los casos de prueba antes del software. Al cabo antes o después hay que hacerlo.
- Pero si lo haces antes, detectas los errores antes que si lo haces después y puedes corregir más fácil.

Pruebas limpias y sucias

- Limpia (L): prueba como funciona el programa cuando debe funcionar.
- Sucia (S): busca romper el programa y observar su comportamiento al romperse.
- Madurez L/S: 5 (inmaduro), ⅕ (maduro, siempre y cuando no sea disminuyendo L).

¿Con qué nos enfrentamos?

```
char    codigoproducto[32];
```

¡¡26^{32} = 1.9 $X10^{45}$ posibles códigos!! (considerando sólo letras del alfabeto)

Prueba exhaustiva

Impensable. Cada variable puede tener una enorme cantidad de posibles valores y las combinaciones de valores de diferentes variables es entonces un número astronómico.

¿Cómo diseñamos la prueba?

White box: tratando de hacer casos que cubran...

- Todas las funciones.
- Todos las decisiones posibles.
- Todos las condicionales.
- Todos los posibles estados de las variables.

James Bach on Software Testing

http://www.youtube.com/watch?v=ILkT_HV9DVU