FACULTAD DE INGENIERIA

Ingeniería de Software

Verificación & Validación

Sommerville capítulo 8
Spillner capítulos 1, 2, 3, 4 y 5
Generating Test Cases From Use Cases

Temario

1) Terminología de verificación y validación
2) Definiciones y principios del testing
3) Clasificación de defectos
4) Verificación y validación en el cliclo de vida del software
    4.1) Modelo en V y niveles de pruebas
    4.2) Pruebas de componentes
    4.3) Pruebas de integración
    4.4) Pruebas de sistema

Terminología (1)

Errores, faltas, defectos y fallas...

Un error humano

Puede provocar

Falta o defecto (interna)

Puede provocar

Falla (externa)

Fallas del software

- ¿El software falló?
  - No hace lo requerido (o hace algo que no debería).
- Algunos motivos:
  - Las especificaciones no estipulan exactamente lo que el cliente precisa o quiere (reqs. faltantes o incorrectos).
  - Requerimiento no se puede implementar.
  - Defectos en el diseño.
  - Defectos en el código.
- **Objetivo**: detectar y corregir estos defectos antes de liberar el producto.

Terminología (2)

- *Testing* (probar) no es *debugging*
  - Probar (*testing*) es una simple revisión del comportamiento del software mediante su ejecución.
  - *Debugging* es la actividad de localizar el defecto en el código y corregirlo.
- Algunos **objetivos** de Verificación y Validación:
  - Ejecutar el programa para provocar fallas (una forma de encontrar defectos).
  - Ejecutar el programa para medir su calidad.
  - Ejecutar el programa para generar confianza.
  - Analizar/revisar el programa o su documentación para detectar defectos (y prevenir fallas).

Terminología (3)

- Objetivo de prueba o tipo de prueba
- Técnica de prueba
- Objeto de prueba
- Nivel de prueba
- Perfil de persona que prueba
- Grado de prueba
- Bases de prueba
- Caso de prueba

(Ref: Sección 2.1.2 Spillner)

Definición de Verificación y Validación

- Verificación: ¿estamos construyendo el producto correctamente?
  - Busca comprobar que el sistema cumple con los requerimientos especificados (funcionales y no funcionales).
  - ¿El software está de acuerdo con su especificación?
  - Testing de defectos.
- Validación: ¿estamos construyendo el producto correcto?
  - Busca comprobar que el software hace lo que el usuario espera.
  - ¿El software cumple las expectativas del cliente?
  - Testing de validación.

Principios de la verificación y validación

1. El testing muestra la presencia de defectos, no la ausencia de ellos
2. Testing exhaustivo no es posible
3. Las actividades de verificación deberían comenzar lo antes posible
4. Agrupamiento/aglomeración de defectos (*defect clustering*)
5. La paradoja del pesticida
6. El testing es contexto-dependiente
7. Decir que el software es útil porque “no se experimentan falllas” es una falacia

Ejemplo

- El programa:
  - Lee tres números enteros, los que son interpretados como representaciones de las longitudes de los lados de un triángulo.
  - Escribe un mensaje que informa si el triángulo es escaleno, isósceles o equilátero.
- Quiero detectar defectos probando (testeando) el programa
- ¿Qué puedo probar?

Ejemplo (cont.)

- Posibles casos a probar:
  - lado1 = 0, lado2 = 1, lado3 = 0 Resultado = error
  - lado1 = 2, lado2 = 2, lado3 = 3 Resultado = isósceles
- Intuitivamente que otros casos sería bueno probar
  - lado1 = 2, lado2 = 3, lado3 = 4 Resultado = escaleno
  - lado1 = 2, lado2 = 2, lado3 = 2 Resultado = equilátero
- ¿Porqué estos casos?
  - Al menos probé un caso para cada respuesta posible del programa (error, escaleno, isósceles, equilátero)
  - Más adelante veremos técnicas para seleccionar casos interesantes

Ejemplo (cont.)

- Otra forma para detectar defectos es revisar el código

```
If I1 = I2 or I2 = I3 then
        write ("equilátero")
else ...
```

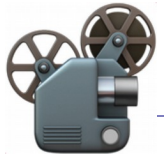
  - En lugar del "or" me doy cuenta que debería ir un "and"
  - Se puede revisar solo o en grupos

Algunos defectos...

- en algoritmos
- de sintaxis
- de precisión y cálculo
- de documentación
- de estrés o sobrecarga
- de capacidad o de borde
- de sincronización o coordinación
- de capacidad de procesamiento o desempeño
- de recuperación
- de estándares y procedimientos
- relativos al hardware o software del sistema

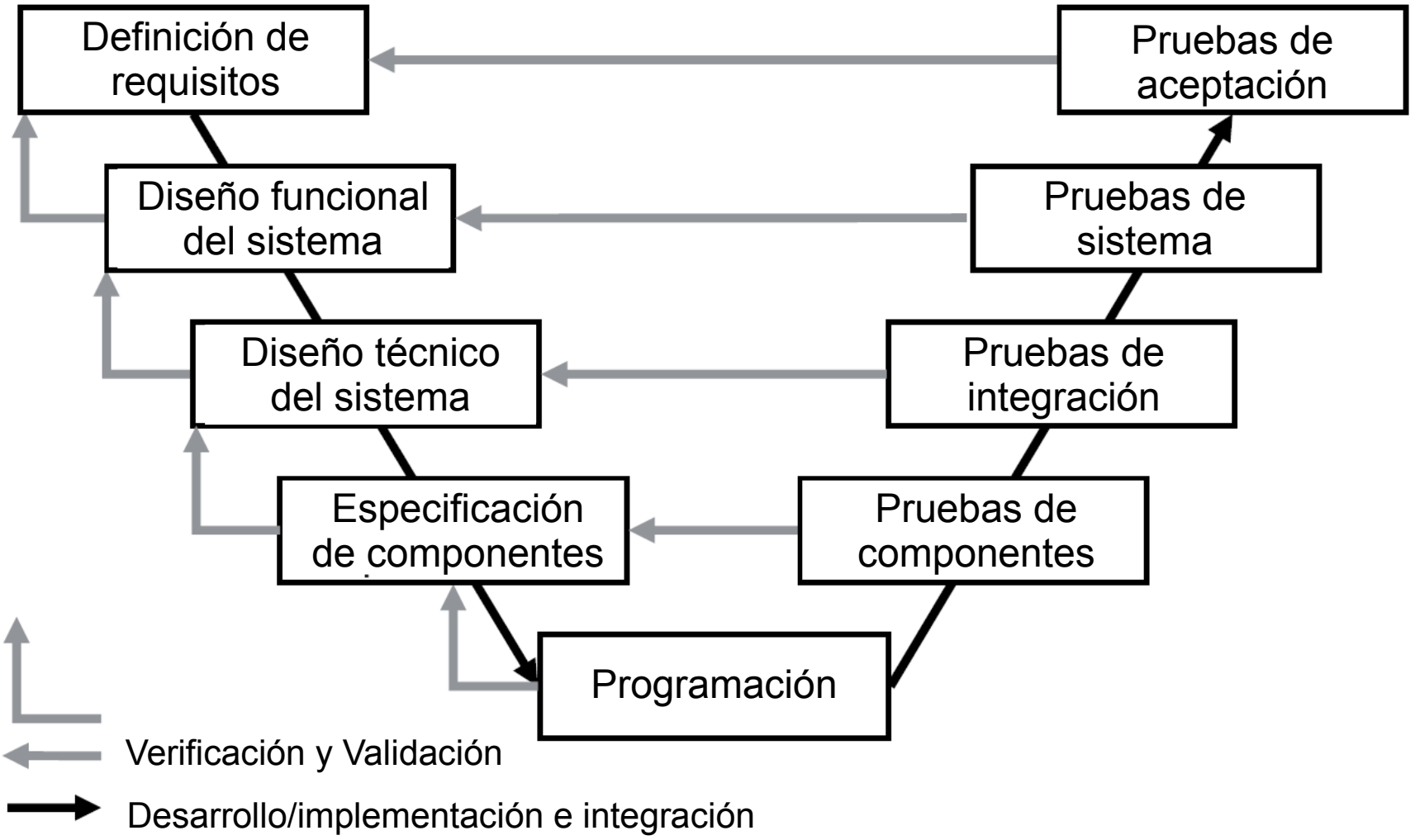
Clasificación de defectos

- Categorizar y registrar los tipos de defectos
  - Guía para orientar la verificación.
    - Si conozco los tipos de defectos en que incurre el equipo de desarrollo me puedo ocupar de buscarlos expresamente
  - Mejorar el proceso
    - Si tengo identificada la fase en la cual se introducen muchos defectos me ocupo de mejorarla
- Clasificación Ortogonal
  - Por ejemplo: tipo de defecto, fuente, impacto, disparador, severidad/ criticidad.



Verificación y validación en el ciclo de vida del Software

Proceso en V


Definición de requisitos
Diseño funcional del sistema
Diseño técnico del sistema
Especificación de componentes
Programación
Pruebas de componentes
Pruebas de integración
Pruebas de sistema
Pruebas de aceptación
Verificación y Validación
Desarrollo/implementación e integración

Niveles de prueba

- **De módulo, componente o unitaria**: verifica el funcionamiento de los componentes de acuerdo a su especificación
- **De integración:** verifica que los grupos de componentes interactúan de la forma en la cual fue especificada acorde al diseño técnico del sistema.
- **De sistema**: determina si el sistema integrado (como un todo) cumple con los requisitos especificados.
- **De aceptación**: verifica que el sistema cumple con los requisitos del cliente de acuerdo a como los mismos fueron especificados en el contrato y/o el sistema cumple con las necesidades y expectativas del cliente.

¿Quién verifica?

- Pruebas Unitarias
  - Normalmente las realiza el equipo de desarrollo. En general la misma persona que lo implementó.
  - Es positivo el conocimiento detallado del módulo a probar
- Pruebas de Integración
  - Normalmente las realiza el equipo de desarrollo
  - Es necesario el conocimiento de las interfaces y funciones en general
- Resto de las pruebas
  - En general un equipo especializado (verificadores)
  - Es necesario conocer los requerimientos y tener una visión global

Depende del proceso y el equipo

¿Quién Verifica?

- ¿Por qué un equipo especializado?
  - Maneja mejor las técnicas de pruebas
  - Conoce los errores más comunes realizados por el equipo de programadores
  - Conoce el modelo de negocio
  - Problemas de psicología de pruebas
    - El autor de un programa tiende a cometer los mismos errores al probarlo
    - Debido a que es “SU” programa inconscientemente tiende a hacer casos de prueba que no hagan fallar al mismo
    - Puede llegar a comparar mal el resultado esperado con el resultado obtenido debido al deseo de que el programa pase las pruebas