Técnicas basadas en la experiencia

Técnicas basadas en la experiencia

- No es un enfoque sistemático (como vimos anteriormente), sino que por el contrario, se basa fuertemente en la intuición, habilidades, conocimiento y experiencia del tester.
- Los casos de prueba se basan en datos históricos (errores/defectos del pasado) o en la intuición de dónde podrían estar en el futuro (*error guessing*).
- Si no existe documentación para utilizar como bases del testing (requisitos, casos de uso, etc.), como técnica puede utilizarse el testing exploratorio.

Testing exploratorio (cont.)

- 3 actividades de forma simultánea:
  - Aprendizaje/exploración de la aplicación
  - Diseño de casos de prueba
  - Ejecución de casos de prueba
- Si bien no hay actividades explícitas de planificación
  - Se puede utilizar el enfoque basado en sesiones
  - Definir el objetivo que queremos cumplir con la sesión de testing
    - Establecer flujos de la aplicación
    - Ver cómo se integra con otra aplicación
    - Robustez
  - Limitar el tiempo dedicado a dicha sesión

Testing exploratorio

- Los elementos del software bajo prueba son “explorados”
  - En una primera instancia se ejecutan unos pocos casos de prueba para obtener conocimiento acerca del software.
  - Se deriva el comportamiento del software (qué es y cómo funciona, qué problemas de calidad podría tener y qué expectativas debería cubrir).
  - En base a los resultados de sesiones anteriores se planifican nuevas sesiones utilizando una “hoja de ruta” para cada una.
- Esta técnica es altamente dependiente de las habilidades del tester
  - Pensamiento crítico, observadores, analíticos
  - Conocimiento del negocio del software

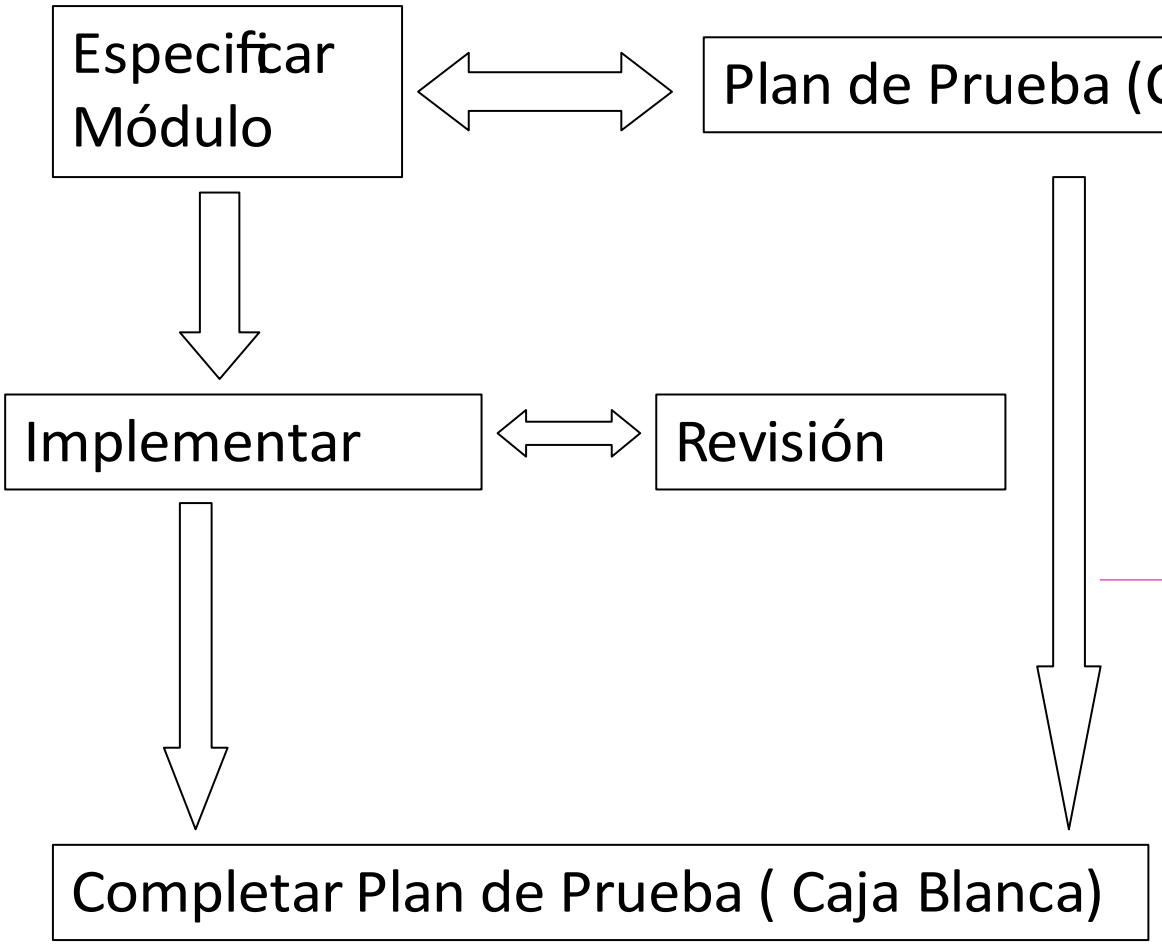
Testing exploratorio (cont.)

- Preguntas importantes a realizarse en cada sesión de prueba:
  - ¿Cuál es el objetivo?
  - ¿Qué debe ser probado?
  - ¿Qué método de prueba debo usar?
  - ¿Qué tipo de problemas podría encontrar?
- Consideraciones generales:
  - Los resultados de los CP ejecutados influencian el diseño y ejecución de los CP que siguen a continuación
  - Durante las pruebas se construye un modelo mental del SW
  - Las pruebas se ejecutan contra ese “modelo” mental

Resumiendo...

- Vimos una amplia variedad de técnicas de prueba... ahora bien, ¿cuáles debería usar?
- Factores que influencian:
  - El tipo de objeto de prueba
  - Documentación formal y/o disponibilidad de herramientas
  - Cumplimiento de estándares
  - Experiencia de los testers
  - Necesidades/deseos del cliente
  - Evaluación de riesgos

Proceso para un módulo (sugerencia)



Jacobson sugiere ejecutar primero las pruebas de caja negra y cuando estas sean todas correctas completar con caja blanca. Esto se debe a que la caja blanca depende del código y cuando se detecten errores con caja negra el código cambia al corregir los errores detectados.

Comparación de técnicas

- Estáticas

✓ Efectivas en la detección temprana de defectos

✓ Sirven para verificar cualquier producto (requerimientos, diseño, código, casos de prueba, etc)

✓ Conclusiones de validez general

✗ Sujeto a los errores de nuestro razonamiento

✗ No se usan para validación (solo verificación)

✗ No consideran el hardware o el software de base

- Dinámicas

✓ Se considera el ambiente donde es usado el software (realista)

✓ Sirven tanto para verificar como para validar

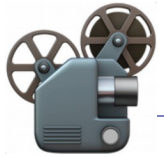
✓ Sirven para probar otras características además de funcionalidad

✗ Está atado al contexto donde es ejecutado

✗ Su generalidad no es siempre clara (solo se aseguran los casos probados)

✗ Solo sirve para probar el software construido

✗ Normalmente se detecta un único error por prueba (los errores cubren a otros errores o el programa colapsa)

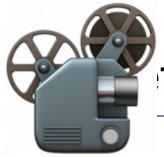


Pruebas no funcionales (desempeño)

- Lo esencial es definir:
  - Procedimientos de prueba
    - Ejemplo: Tiempo de respuesta
      - Simular la carga, tomar los tiempos de tal manera, número de casos a probar, etc
- Criterios de aceptación:
  - Ejemplo: El cliente lo valida si en el 90% de los casos de prueba en el ambiente de producción se cumple con los requerimientos de tiempo de respuesta
- Características del ambiente
  - Definir las características del ambiente de producción ya que estas hacen grandes diferencias en los requerimientos no funcionales

Tipos de pruebas no funcionales

- Estándares internacionales de pruebas acorde a los atributos de calidad del software: ISO 9241, ISO 9126,ISO/IEC 25010:2011.
  - Prueba de carga
  - Prueba de performance (rendimiento)
  - Prueba de volumen
  - Prueba de estrés
  - Prueba de seguridad
  - Prueba de confiabilidad
  - Prueba de robustez
  - Prueba de compatibilidad y conversión de datos
  - Prueba de configuración
  - Prueba de facilidad de uso
  - Comprobación de documentación



f: Sección 3.7.2 Spillner

Gestión de las pruebas (*Test managment*)

Spillner secciones 6.1 y 6.2

Organización del testing

- Así como en el desarrollo, las actividades de testing necesitan ser organizadas. Las mismas deben ser coordinadas con las actividades de desarrollo.
- Ya hablamos de los beneficios de que el testing sea independiente del desarrollo (sicología de las pruebas). Algunas opciones de organización serían:
  - El equipo de desarrollo es el responsable de las pruebas, pero los desarrolladores se verifican mutuamente el código.
  - Hay roles específicos de testers dentro del equipo de desarrollo.
  - Uno o más equipos de testing dedicados al proyecto trabajan en conjunto con el equipo de desarrollo.
  - Especialistas independientes en pruebas se encargan de algunas actividades específicas de testing (usabilidad, seguridad, estándares).
  - Tercerización del testing en una organización separada.

Roles en las pruebas

- Líder de testing (*test manager*): planifica y da seguimiento a las actividades de verificación.
  - Selecciona y da soporte a las estrategias de pruebas a utilizar, procura y asigna los recursos a las mismas.
  - Identifica las métricas necesarias para dar seguimiento al progreso de las pruebas y controlar la calidad del producto de software.
- Diseñador de pruebas (test analyst): experto en los métodos/técnicas de prueba.
- Automatizador de pruebas: tiene conocimiento de testing así como también de programación y herramientas de automatización.
- Administrador de pruebas: expertos en instalar y operar el entorno de pruebas
- Probador (*Tester*): ejecuta y reporta los resultados de las pruebas

Planificar la verificación

- Las actividades de verificación se deben planificar en mayor o menor grado, dependiendo y siendo coherente con el enfoque del proceso de desarrollo.
- Permite que el personal técnico obtenga una visión global de las pruebas del sistema y ubique su propio trabajo en ese contexto.

Test Plan according to IEEE 829-1998

1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria (test exit criteria)
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing and training needs
14. Schedule
15. Risk and contingencies
16. Approvals

Planificar la verificación (cont.)

- La V&V es un proceso caro se requiere llevar una planificación cuidadosa para obtener el máximo provecho de las revisiones y las pruebas para controlar los costos del proceso de V&V
- El proceso de planificación de V&V
  - Pone en la balanza los enfoques estático y dinámico para la verificación y la validación
  - Utiliza estándares y procedimientos para las revisiones y las pruebas de software
  - Establece listas de verificación para las inspecciones
  - Define el plan de pruebas de software

Planificar la verificación (cont.)

- El mayor error cometido en la planificación de un proceso de prueba:

  Suponer que no se encontrarán fallas al momento de realizar el cronograma

- Los resultados de esta equivocación son obvios
  - Se subestima la cantidad de personal a asignar
  - Se subestima el tiempo de calendario a asignar
  - Entregamos el sistema fuera de fecha o ni siquiera entregamos