Pruebas de software

Luis Daniel Benavides Navarro, Ph.D.

Introducción a pruebas de software

- Casos de estudio: errores famosos
- Que son las pruebas de software
- Definiciones
- Los limites de la pruebas
- El trabajo del ingeniero de pruebas

Casos de estudio: errores famosos

- Lion King (1994-1995)
- Intel pentium. División de punto flotante (1994)
- Nasa: Mars Polar Lander (1999)
- Misil de defensa patriot (1991)
- Y2K (70s)

Errores famosos: Lion King (1994-1995)

- Cuando: Otoño del 1994
- Producto: Juego CD-ROM Multimedia "The Lion King animated Storybook"

- Error: El juego no funcionaba.
- Efecto: Línea de atención al cliente congestionada el 26 de Dic. (imagén franquicia)
- Origen del problema: Disney no probó el juego en una amplia gama de PCs.

Errores famosos: Intel pentium. División de punto flotante (1994)

- Cuando: 1994
- Producto: Intel pentium
- Error: El resultado de (4195835 / 3145727) * 3145727 - 4195835 era diferente a 0.
- Efecto: 400 millones de dólares para recoger equipos. Mala imagen franquicia.
- Origen del problema:
  - Detectaron el error pero decidieron no corregirlo
  - Ocultaron el error al público
  - Ofrecieron solución pero dificultando acceso a la solución

Errores famosos:
Nasa, Mars Polar Lander (1999)

- Cuando: 1999
- Producto: Mars Polar Lander
- Error: hipótesis, los propulsores se desactivaron a 1800m de la superficie
- Efecto: Mars Polar Lander desapareció durante el aterrizaje
- Origen del error:
  - Cambio del sistema sensor de altura para cortar el flujo a propulsores por un interruptor mecánico en las piernas de aterrizaje (activaba un bit que cortaba el flujo)
  - Interruptor fue activado por vibración
  - Pruebas de interruptor y sistema de aterrizaje hecha por equipos diferentes

Errores famosos: Misil de defensa patriot (1991)

- Cuando: 1991
- Producto: Misil de defensa patriot (Star-Wars)
- Error: Sistema de rastreo era impreciso
- Efecto: 28 soldados americanos muertos (guerra del golfo)
- Origen del problema:
  - Error acumulativo en el reloj del sistema los hacía impreciso después de 14 horas de operación
  - El día del ataque el sistema tenía más de 100 horas de operación

Errores famosos: Y2K (70s)

- Cuando: en los 70
- Producto: Gran porcentaje del software existente a finales de los 90
- Error: Los sistemas utilizaban dos carcteres para almacenar el año (ej.: 73 y no 1973)
- Efecto: La fecha de los sistemas retrocedería 100 años el 1 de enero del 2000.
- Origen del problema:
  - Problemas conocidos pero programador consideró que 25 años era suficiente tiempo para remplazar el sistema.
  - No se corrigió el error.

Que son las pruebas de software

- Un problema de ingeniería de sistemas
- Es el proceso de diseño e implementación de un sistema para ejercitar otro sistema
- Desarrollo de un sistema automatizado para aplicar pruebas a un sistema
- Pruebas manuales aún juegan un rol

El proceso de diseño de pruebas

- Identificar, modelar, y analizar las responsabilidades del sistema bajo pruebas
- Diseñar casos de test basados en esta información externa
- Adicionar casos de test basado en análisis de código, experiencia y heurísticas
- Crear resultados de pruebas para cada caso y elegir un estrategia para determinar si éxito o fracaso para cada caso de prueba

Los limites de la pruebas: El espacio de entrada y salida

- Las combinaciones de entrada y salida para programas triviales son muy grandes
- Para los sistemas están fuera de la comprensión
- En el programa del triángulo, en una pantalla de 1024x768 existen 786.432^6 casos de prueba.
- Probando 1000 casos por segundo 7x24 usted necesitaría más de la edad del universo para probar todos los casos de prueba.

Los limites de la pruebas: Secuencias de ejecución

- Tomado de [Binder2000]

```
for(in i = 0; i < n;  i++){
        if(a.get(i) == b.get(i))
            x[i] = x [i] +100;
        else
            x[i] = x [i] *2;
}
```

- Si n = 1 hay 3 posibles secuencias de ejecución
- Si n = 10 hay 1025 posibles secuencias de ejecución
- Si n = 20 hay 1.048.577. posibles secuencias de ejecución
- 2^n +1 caminos posibles de ejecución

Los limites de la pruebas: Sensibilidad a fallas y resultados correctos coincidenciales

```
public class Account{

    Date lastTxDate, today;

    int quarterSinceLastTx(){

        return (90 / daysSinceLastTx());

    }

    int daysSinceLastTx(){

        return (today-day() - lastTxDate.txDay +1);

    }

}
```

```
public class TimeDepositAccount extends Account{

    ...

    int daysSinceLastTx(){

        return (today-day() - lastTxDate.txDay);

    }

}
```

- Capacidad de ocultar errores
- Ejemplo: polimorfismo en OO

Los limites de la pruebas: Límites absolutos

- Pruebas no pueden usarse para demostrar la ausencia de defectos
- Imposible automatizar las pruebas de ciertas propiedades del software, e.g., El problema de la parada.
- Pruebas están basadas en los requerimientos redactados subjetivamente.
- Pruebas basadas en la implementación no puede probar omisión de código
- No se esta seguro que el sistema de pruebas sea correcto
- No se tiene absoluta certeza sobre los resultados utilizados para diseñar los casos de prueba.

El trabajo del ingeniero de pruebas

- Encontrar bugs, lo más temprano posible, y asegurarse de que queden arreglados
- No es un substituto de las buenas prácticas de ingeniería de software
- Probar todos artefactos desde las etapas iniciales del proceso de desarrollo
- Paso necesarios para convertir prototipos en productos

Probando a través del ciclo de desarrollo

- Etapas comunes de los ciclos de desarrollo
- Servicios transversales a los ciclos de desarrollo
- Modelos de madurez y calidad
- Ejemplos de ciclos de desarrollo específicos
- Niveles de pruebas
- Tipos de pruebas
- Pruebas de mantenimiento

Etapas comunes en los modelos de desarrollo de software

- Especificación
- Diseño abstracto (diseño)
- Diseño concreto (implementación)
- Pruebas
- Despliegue

Servicios transversales a los ciclos de desarrollo

- Manejo de configuraciones
  - Versiones de código fuente
  - Versiones de documentos
    - Relaciones de documentos, e.g., casos de uso vs. requerimientos, requerimientos vs. diseño, req. vs casos de prueba
- Configuración de ambientes
- Pruebas? (¿qué opinan?)

Modelos de Madurez y Calidad

- CMMI, ISO, ITIL, COBIT
- Los modelos CMMI (Capability Maturity Model Integration) son colecciones de mejores prácticas que sirven para mejorar los procesos de las organizaciones.
- Niveles de madurez propuestos
  - Inicial: Procesos caóticos
  - Administrado: Planeación, ejecución, medición y control. No- repetible.
  - Definido: Procesos definidos, claros con herramientas y prácticas definidas.
  - Administrado cuantitativamente: Objetivos cuantitativos para la calidad y el proceso son definidos y usado para la administración
  - Optimizando: La efectividad del proceso es mejorada continuamente pro medio de mejoras incrementales.

El modelo big-Bang

- Disponibilidad de recursos
- No espacio para el test formal
- Generalmente solo pruebas de producto final
- Proceso de caja negra

El modelo Code-and-Fix

- Especificación informal del producto
- Diseño abstracto informal
- Ciclos de codificar, probar, arreglar
- Buen esquema de prototipos

El modelo de cascada

- Etapas claras de proceso de software
- Etapas en secuencia
- Etapa formal de pruebas
- Enfasis en la especificación
- Generalmente pruebas demasiado tarde en el proceso

Modelo espiral

- No define todo en detalle al principio
- Empieza pequeño, hace un cliclo de desarrollo, obtiene retroalimentación
- En cada iteración agranda el alcance del proyecto
- Pruebas en cada ciclo
- Oportunidad de pruebas tempranas para influenciar el desarrollo

Agiles iterativos

- agilemanifesto.org

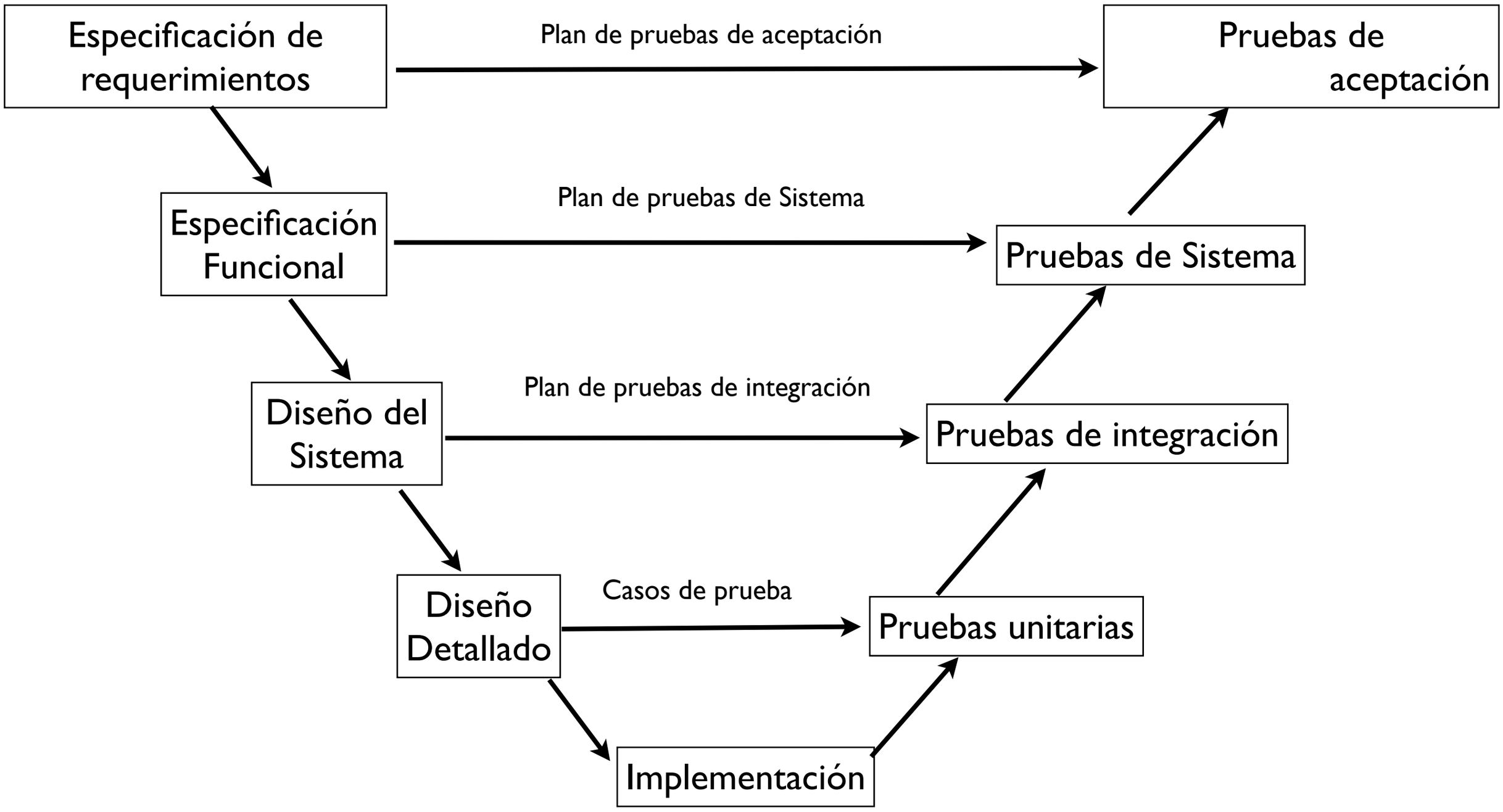
  "We are uncovering better ways of developing
  software by doing it and helping others do it.
  Through this work we have come to value:

  Individuals and interactions over processes and tools
  Working software over comprehensive documentation
  Customer collaboration over contract negotiation
  Responding to change over following a plan"

  That is, while there is value in the items on
  the right, we value the items on the left more."

- Iterativos
- Contacto constante con el usuario
- Enfocado en software funcionando

El modelo V


Especificación de requerimientos
Plan de pruebas de aceptación
Pruebas de aceptación
Especificación Funcional
Plan de pruebas de Sistema
Pruebas de Sistema
Diseño del Sistema
Plan de pruebas de integración
Pruebas de integración
Diseño Detallado
Casos de prueba
Pruebas unitarias
Implementación

Niveles de pruebas

- Pruebas de componente
- Pruebas de integración
- Pruebas de sistema
- Pruebas de Aceptación

Pruebas de componentes (e.g, unitarias)

- Pruebas basadas en:
  - Requerimientos de componentes
  - Diseño detallado
  - Código
- Objetos bajo prueba típicos:
  - Componentes
  - Programas
  - Programas de conversión / migración de datos
  - Módulos de la base de datos

Pruebas de integración

- Pruebas basadas en:
  - Diseños de software y sistema
  - Arquitectura
  - Flujos de trabajo
  - Casos de uso
- Objetos bajo prueba típicos:
  - Implementaciones de bases de datos de subsistemas
  - Infraestructura
  - Interfaces
  - Configuración del sistema
  - Datos de configuración

Pruebas de sistema

- Pruebas basadas en:
  - Especificación de requerimientos de sistema y software
  - Casos de Uso
  - Especificación funcional
  - Reportes de análisis de riesgo
- Objetos bajo prueba típicos:
  - Sistema, manuales de usuario y operación
  - Configuración del sistema
  - Datos de configuración

Pruebas de aceptación

- Pruebas basadas en:
  - Requerimientos de usuario
  - Requerimientos de sistema
  - Casos de uso
  - Procesos de negocio
  - Análisis de riesgo
- Objetos bajo prueba típicos:
  - Procesos de negocio en el sistema totalmente integrado
  - Procesos de operación y mantenimiento
  - Procedimientos de usuairo
  - Formatos
  - Reportes

Tipos de pruebas

- Pruebas Funcionales
- Pruebas no-funcionales
- Pruebas de arquitectura / estructura
- Re-testing y pruebas de regresión

Pruebas funcionales

- Funciones o características del sistema, componente o módulo
- Considera el funcionamiento externo del objeto bajo prueba
- Son derivadas con técnicas basadas en la especificación.

Pruebas no funcionales

- Características no funcionales como desempeño, disponibilidad, carga, estrés, usabilidad, cofiabilidad
- Considera el comportamiento externo del sistema bajo prueba
- Son derivadas de la especificación y estándares bien documentados

Pruebas de estructura/ arquitectura

- Estructural, caja blanca, efectuada a todos los niveles
- Ayuda a evalúar como ha sido el cubrimiento de las pruebas,i.e., que tanto ha sido ejercitado el sistema
- Son derivadas de documentos de estructura, estándares de codificación, especificaciones de GUI entre otros

Pruebas relacionadas a cambios, pruebas de regresión

- Después de que un defecto es detectado y arreglado
- Las pruebas de regresión garantizan que bugs viejos no aparezcan y no perder funcionalidad ya probada
- Un buen candidato para automatización

Pruebas de mantenimiento

- Realizadas sobre un software en operación
- Disparadas por el cambio
  - Modificaciones
  - Migración
  - Retiro del software
- También debe considerar cambios inesperados y hacer pruebas de regresión
- El éxito depende de la anticipación al cambio, e.g., tamaño de base de datos, Y2K, servicios externos

El ciclo de pruebas

- Planeación y control
- Análisis y diseño
- Implementación y ejecución
- Evaluando el criterio de salida y reportando
- Actividades de cierre

Derivación de pruebas

- Un técnica de derivación de pruebas debe encontrar:
  - condiciones de prueba
  - casos de prueba
  - datos de prueba
- Técnicas de derivación de prueba
  - Caja Negra
  - Caja Blanca
  - Experiencia

Pruebas de caja negra

- Derivación de pruebas basado en documentación de especificación
  - Modelos
  - Documentos
  - Diseños
- Incluye pruebas funcionales y no funcionales
- No usa información interna del sistema o componente bajo prueba

Pruebas de caja blanca

- Basadas en el análisis de la estructura del sistema o componente bajo prueba
  - Código
  - Diseño detallado
- Se puede medir la cobertura de los casos de prueba
- Se pueden derivar sistemáticamente casos de prueba para mejorar cobertura

Derivación basada en experiencia

- Basadas en la experiencia e intuición de los sistemas
- Se busca involucrar experiencia de varios stakeholders

Ejecución de pruebas

- Pruebas estáticas
- Pruebas dinámicas

Pruebas estáticas

- Análisis estático de artefactos del proceso de software
- Encuentran causas de fallas
- Realizadas con un objetivo específico
- Algunos ejemplos
  - Peer review
  - Uso del compilador

Pruebas dinámicas

- A diferencia de las estáticas encuentra los fallos mismos
- Se realiza con la ejecución del software