Pruebas

- Pruebas en el PUD
- Las pruebas del software
- Diseño de casos de prueba
- Pruebas de SI OO

Iteración en PUD

- Planificación de la Iteración
  - Captura de requerimientos:
    - Modelo de casos de uso, Modelo de Dominio, ...
  - Análisis:
    - Diagrama de secuencia del sistema, Contratos, Modelo Conceptual...
  - Diseño:
    - Diagramas de interacción, Diagrama de Clases
  - Implementación:
    - codificación (Clases y métodos)
  - Pruebas:
    - verificación de la implementación
- Evaluación de la iteración

Fases y entregas del Proceso Unificado de Desarrollo

- captura de requerimientos: qué SI debemos construir?
  - Modelo de casos de uso, Modelo de Dominio, ...
- análisis: qué debe hacer el SI?
  - Diagramas de secuencia del sistema, Contratos, ...

- diseño: cómo lo debe hacer el SI?
  - Diagramas de interacción, Diagrama de Clases, ...
- codificación:
  - Código Fuente (clases y métodos)
- pruebas:
  - Especificación de las pruebas de funcionamiento
- mantenimiento:
  - Documentación y revisión de todo lo anterior

Dependiente de la tecnología

Las pruebas de software

- Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación

- Las pruebas de software son siempre necesarias

- En algunos casos ocupan un 40% del tiempo de un proyecto informático

- Las pruebas pretenden descubrir errores!

Las pruebas de software

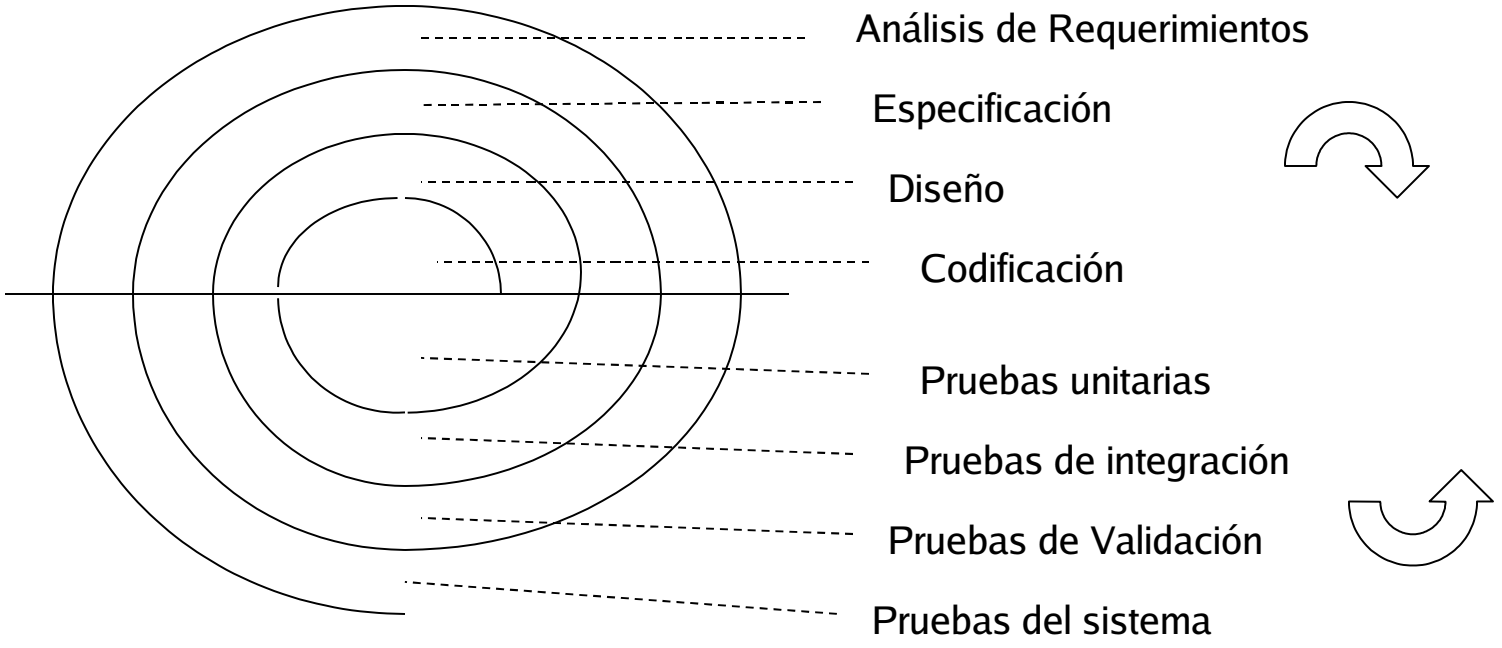
- Un buen caso de prueba es aquel que tiene una probabilidad muy alta de descubrir un nuevo error
- Una prueba tiene éxito si descubre un error nuevo!
- Debemos diseñar y ejecutar juegos de prueba que, de forma sistemática, detecten distintos tipos de error en el menor tiempo y esfuerzo posible
- Los juegos de prueba no deben ser ni excesivamente simples ni exageradamente complejos
- Las pruebas pueden demostrar la existencia de errores, pero no su ausencia!

Las pruebas de software

- Las pruebas pueden planificarse mucho antes de que empiecen
- Empezar por lo pequeño y progresar hacia lo grande
- No son posibles las pruebas exhaustivas!
- Son más efectivas las pruebas dirigidas por un equipo independiente
- El 80% de los errores está en el 20% de los módulos
- Hay que identificar ésos módulos y probarlos muy bien.

Las pruebas de software

- El proceso de IS se puede ver como un proceso en espiral
- El proceso de pruebas de software también, pero en sentido contrario



Las pruebas de software

- Pruebas unitarias
  - Prueba de un único comportamiento elemental
- Pruebas de integración
  - Prueba de las interacciones entre componentes del sistema
  - Verificación incremental
    - Descendente
    - Ascendente
    - Regresión para detectar errores en componentes ya probados!
- Pruebas de validación
  - Se centran en asegurar que se satisfacen los requisitos desde el punto de vista del usuario
- Pruebas del sistema
  - Prueba global del sistema como unidad de ejecución

Diseño de casos de prueba

- Diseñar un juego de pruebas: conjunto de casos de prueba
- Un caso de prueba especifica:
  - Componente a probar
  - Datos de entrada
  - Estado del componente
  - Información de contexto
  - Resultado esperado
- Un caso de prueba
  - Con alta probabilidad de detectar algún error
  - No debe ser redundante
  - Debe ser representativo
  - Ni muy simple ni muy complejo

Casos de prueba

- Deben contemplar
  - La planificación de la prueba
  - El diseño de los casos de prueba
  - La ejecución de la prueba
  - La evaluación de los casos de prueba

- No debemos olvidarnos de:

  - Pruebas sobre la capa de presentación (ventanas, menús, ratón ...)
  - Pruebas sobre la capa de gestión de datos
  - Pruebas de documentación
  - Pruebas de ayuda

Estrategias de prueba

- Pruebas de “caja negra” (funcionales o de comportamiento)
  - El juego de pruebas se diseña considerando las responsabilidades del componente (“vemos” sólo los requisitos funcionales)
  - Permite detectar errores de:
    - asignación de responsabilidades
    - La interfaz del componente

- Pruebas de “caja blanca” (de implementación)
  - El juego de pruebas se diseña analizando/ejecutando el código del componente
  - Intentan garantizar que todos los caminos de ejecución del programa quedan probados
  - Permite detectar errores de:
    - Flujo de control
    - Estructuras de datos locales
    - ...

Pruebas de Unidad

- Centra la prueba en el componente
- Puede realizarse en paralelo a otros componentes
- Básicamente son pruebas de caja blanca
  - Interfaz
  - Condiciones límite
  - Caminos independientes
  - Caminos de tratamiento de errores
- Se prueban los caminos de control importantes para descubrir errores en el componente
- Debemos simular el “comportamiento” del resto de componentes!

Pruebas de Integración

- Pruebas de integración descendente
- Pruebas de integración ascendente
- Pruebas de regresión
  - Cambios o la introducción de un nuevo componente pueden provocar errores en componentes ya probados!
  - Al realizar cambios en algún componente debemos probar de nuevo los componentes ya probados
  - Se realizan las mismas pruebas para asegurarse que no se han producido cambios colaterales

Pruebas de Validación

- Se llevan a cabo cuando se han terminado las pruebas de integración, el software está ensamblado y se han realizado todas las pruebas de unidad e integración
- La validación se consigue cuando el software funciona según las expectativas del usuario!
- Se realizan una serie de pruebas de caja negra que aseguren que se satisfacen los requisitos
  - Funcionales
  - De rendimiento
  - De documentación
  - Transportabilidad, compatibilidad
  - Recuperación de errores
  - ...

Pruebas de Validación

- Pruebas de aceptación: desarrolladas por el cliente
- Pruebas alfa:
  - Realizadas por el usuario con el desarrollador como observador en un entorno controlado (simulación de un entorno de producción)
- Pruebas beta:
  - Realizadas por el usuario en su entorno de trabajo y sin observadores

Pruebas del Sistema

- Realizado el software, éste debe ponerse en explotación e integrarse en un entorno productivo

- Estas pruebas sirven para verificar que se han integrado adecuadamente todos los elementos del sistema y todos ellos de forma conjunta realizan las funciones apropiadas

- Pruebas de seguridad
- Pruebas de resistencia
- Pruebas de rendimiento
- Pruebas de recuperación

Depuración de errores

- Al realizar pruebas pueden descubrirse errores y éstos deben depurarse

- Depurar errores es extremadamente difícil (sobretodo si se trata de un sistema desconocido!)

  - El error puede ser provocado por un “mal uso” no contemplado en el diseño!
  - Puede ser difícil reproducir las condiciones que lo producen
  - El error aparece de forma intermitente
  - Su corrección requiere cambios substanciales del SI!