Curso de preparación
Tester Certificado
Nivel Básico

Agenda

- Capítulo 0 Introducción
- Capítulo 1 Fundamentos de Pruebas
- Capítulo 2 Pruebas a lo largo de Vida del desarrollo
- Capítulo 3 Técnicas Estáticas
- Capítulo 4 Técnicas de Diseño de Pruebas
- Capítulo 5 Gestión de Pruebas
- Capítulo 6 Herramientas de Soporte para el Proceso de Pruebas

Agenda

Capítulo 2 Pruebas a lo largo del ciclo de vida del software

- 2.1 Modelos de desarrollo de Software
- 2.2 Niveles de pruebas
- 2.3 Tipos de prueba
- 2.4 Pruebas de mantenimiento

2.1: Modelos de desarrollo de software

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo de software

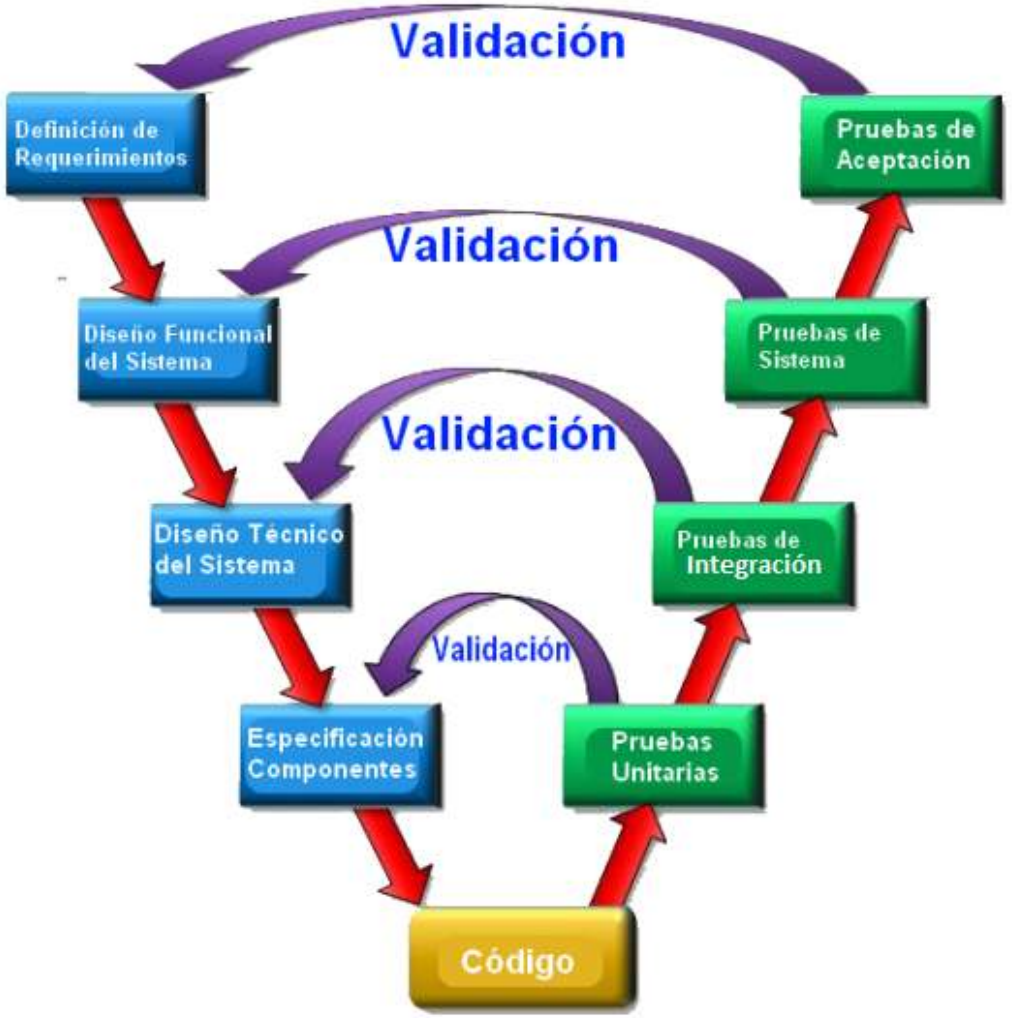
Términos

- **Genérico comercial (COTS - Commercial Off-The-Shelf software):** Producto de software desarrollado para un mercado general.
- **Modelo de desarrollo iterativo-incremental:** Modelo de desarrollo de software que permite la entrega de versiones parciales, incrementándose hasta alcanzar el producto final.
- **Validación:** Confirmación mediante examen y mediante el suministro de evidencia objetiva de que los requisitos para un uso específico previsto o aplicación se han cumplido. [ISO 9000].
- **Verificación:** Confirmación mediante examen y mediante el suministro de evidencia objetiva de que requisitos especificados han sido cumplidos. [ISO 9000].
- **Modelo V:** Un modelo de desarrollo para describir las actividades de desarrollo de software de ciclo de vida desde los requisitos hasta la especificación para mantenimiento. El modelo V ilustra cómo las actividades de prueba se puede integrar en cada fase del ciclo de vida de desarrollo.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

2.1.1. El modelo en V

- El modelo en V de desarrollo de software es el mas utilizado , y aunque existen múltiples variantes siempre utiliza cuatro niveles de pruebas que corresponden a los cuatro niveles de desarrollo, asociados de forma paralela.
- Desarrollo y pruebas son **dos ramas iguales**



http://josepablosarco.wordpress.com/2012/03/24/istqb-cap-2-testing-a-traves-del-ciclo-de-vida-del-software-i/

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

Las pruebas en el modelo en V

- Los niveles del modelo en V de desarrollo pueden presentar variaciones de acuerdo a la naturaleza del proyecto, para el desarrollo de las siguientes lecciones se utilizarán :
  - Pruebas de componente (unitarias).
  - Pruebas de integración.
  - Pruebas de sistema.
  - Pruebas de aceptación.

- Cada uno de los artefactos producidos por las etapas de desarrollo serán el insumo para la ejecución de las pruebas, estos artefactos podrán estar desarrollados en el modelo de madurez de las capacidades integrado (CMMI) o en procesos del ciclo de vida del software (IEEE-IEC 12207)

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

Rama de Software del modelo-V

- **Definición de Requisitos.**
  Documentos de especificación,
  Casos de uso etc.
- **Diseño funcional del sistema.**
  Diseño del flujo funcional del programa.
- **Diseño técnico del sistema.**
  Definición de arquitectura / interfaces.
- **Especificación de los componentes.**
  Estructura de los componentes.
- **Programación.**
  Creación de código ejecutable. Prototipos

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

Rama de Pruebas del modelo-V

- **Pruebas de aceptación.**
  Validar que el sistema cumple con el funcionamiento esperado desde el punto de vista del usuario
- **Pruebas de sistema.**
  Busca validar el comportamiento del sistema de forma global
- **Pruebas de integración**
  Busca validar el funcionamiento Integrado de todos los componentes
- **Pruebas de componente.**
  Funcionalidad del componente de manera individual (unitarias)
- **Programación.**
  Creación de código ejecutable. Prototipos

Verificación y Validación

Verificación

- Comprobación de la conformidad con los requisitos establecidos (definición según ISO 9000).
- ¿He hecho el software correctamente?
- Ejemplo: ¿El calculo de la edad de un paciente es igual a la fecha actual menos la fecha de nacimiento del paciente?

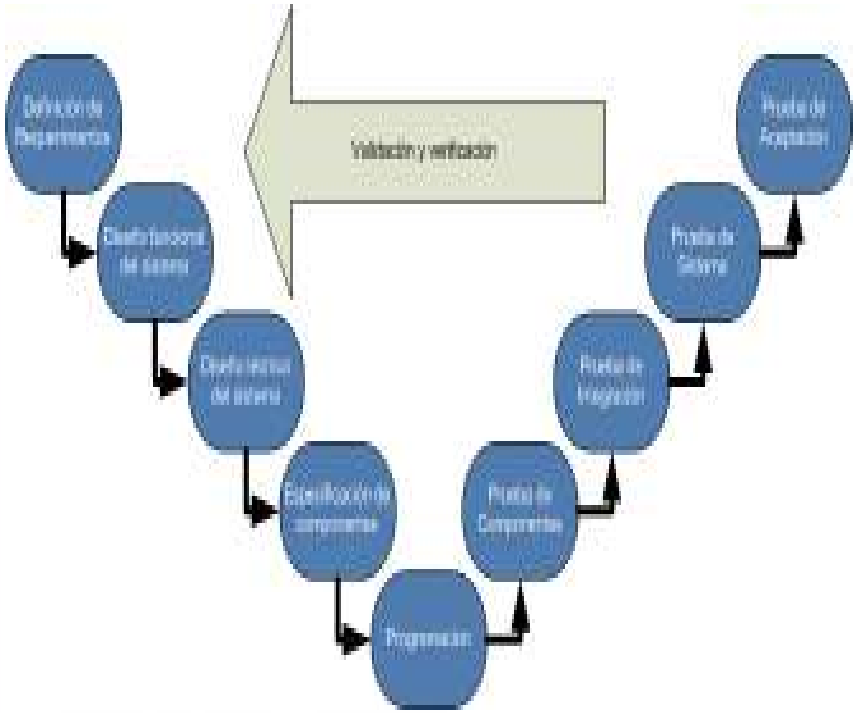
Validación

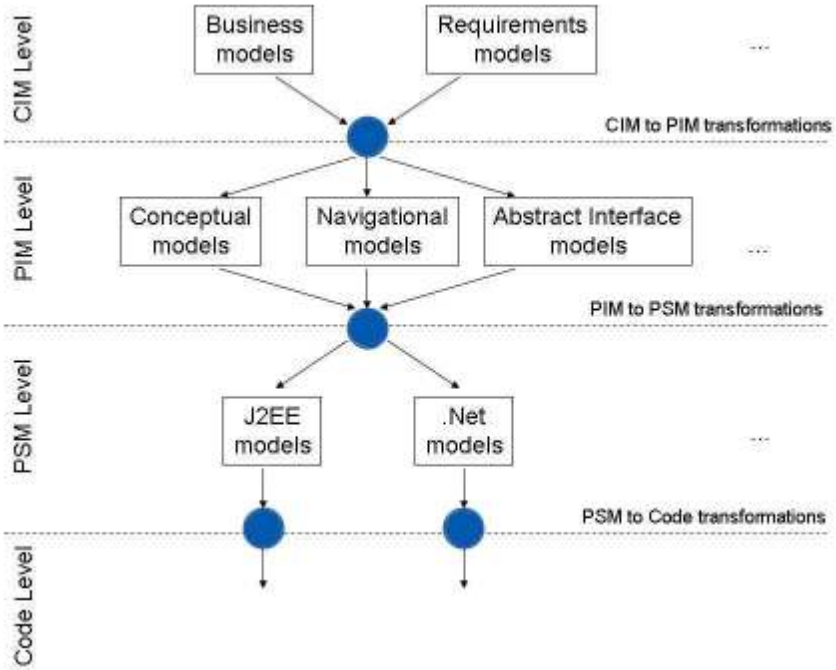
- Comprobación de la idoneidad para el uso esperado (definición según ISO 9000).
- ¿He hecho el software correcto?
- Ejemplo: ¿Realmente necesito la edad del paciente?

Verificación y Validación en el modelo-V

Verificar significa comprobar si los requisitos y definiciones de niveles previos han sido implementados correctamente.

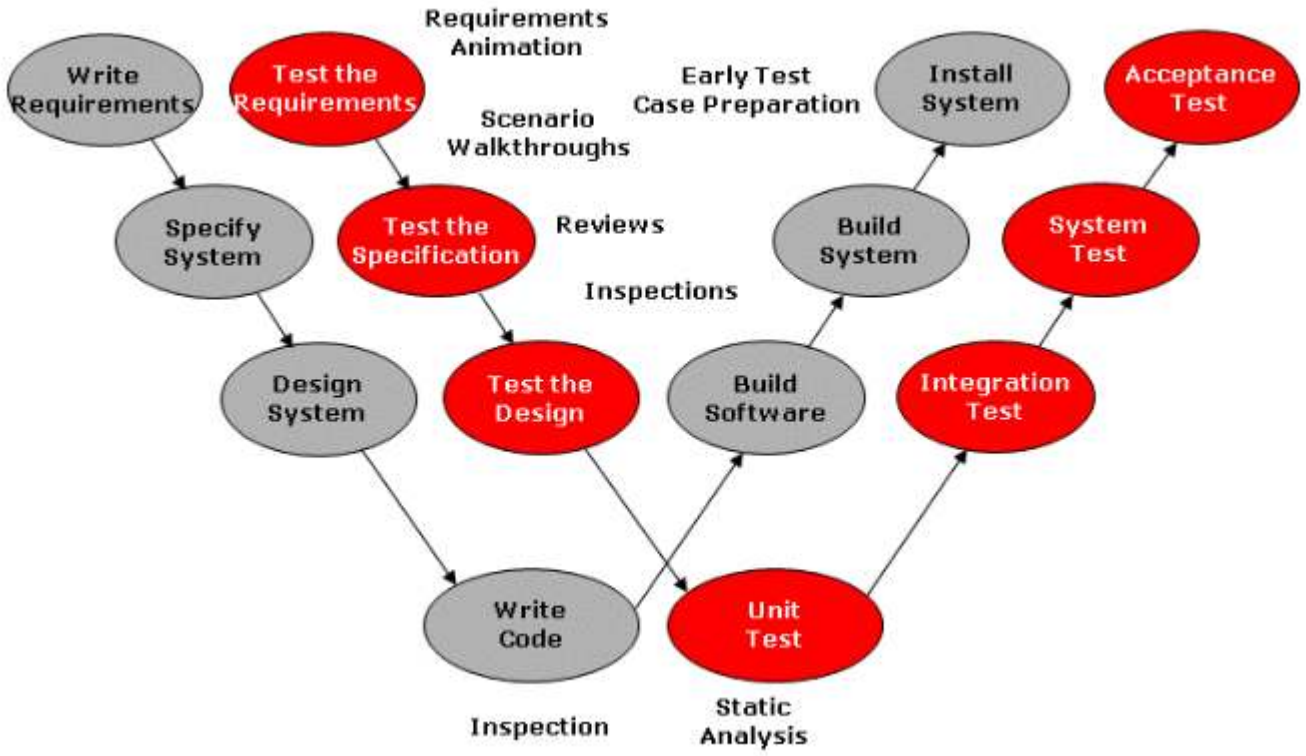
Validar significa comprobar lo adecuado de los resultados de un nivel de desarrollo.





EL modelo W de pruebas de software

- Una de las variaciones al modelo en V presentada por Paul Herzlich, que busca aclarar algunos conceptos que no son tan evidentes.
- Muestra las actividades de pruebas paralelas a las actividades de desarrollo, siendo mas completo que el modelo en V tradicional.



2.1.2 Modelos de desarrollo iterativo-incremental

Definición

- Son variaciones al modelo en cascada, aplicados de forma repetitiva, bien sea por ciclos o por iteraciones.
- Son sistemas con tendencia a crecer con cada ciclo que es ejecutado, dándole mayor importancia a las pruebas de regresión .

Consideraciones

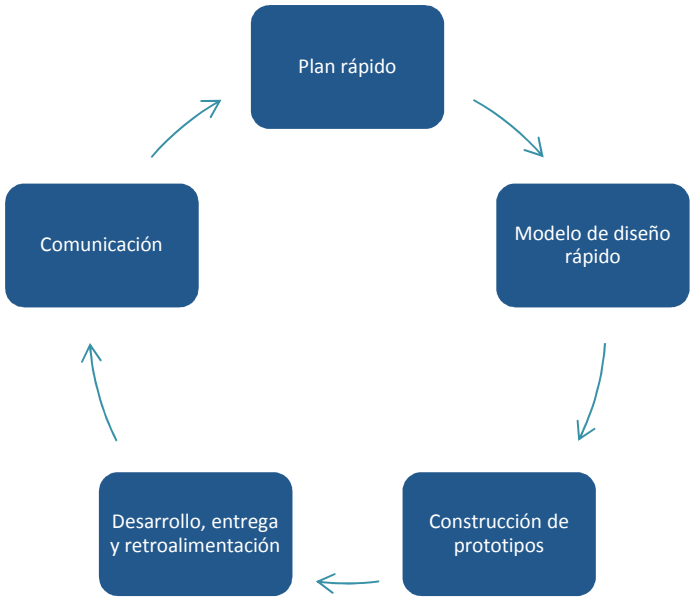
- La validación y verificación se podrán llevar a cabo al finalizar cada uno de los ciclos.
- Se hacen indispensables las pruebas de regresión posteriores a cada incremento o iteración.

Algunos modelos de desarrollo iterativo-incremental

- **Modelo prototipado:** Desarrollo acelerado de prototipos que no utiliza muchos recursos, estos prototipos son evaluados para retroalimentar a un siguiente ciclo.
- **Desarrollo rápido de aplicaciones ("Rapid Application Development" - (RAD)):** Desarrollo de aplicaciones en tiempos cortos, generalmente apoyados con la utilización de herramientas CASE
- **Proceso unificado ("Rational Unified Process" - (RUP)):** Modelo desarrollado por Rational/IBM. Orientado al análisis, implementación y documentación de sistemas orientados a objetos.
- **SCRUM:** Desarrollo ágil de aplicaciones que hace énfasis en la adaptabilidad mas que en la previsibilidad.

Ejemplos de modelos iterativos:

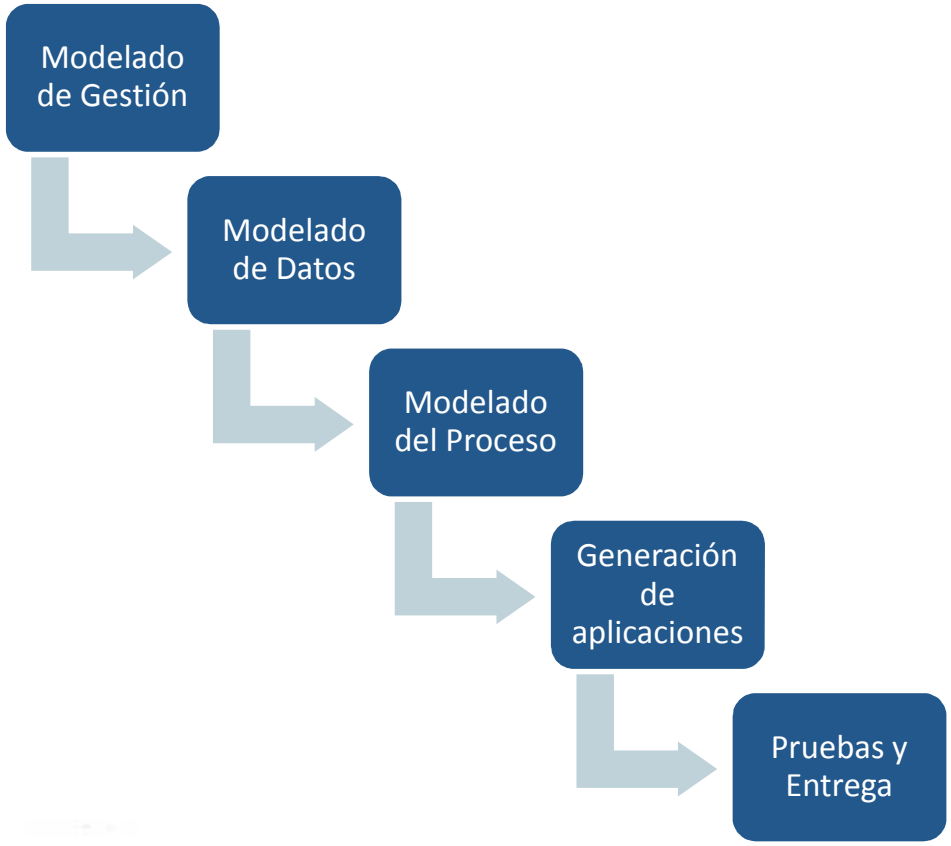
- **Modelo prototipado:** El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos.



El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final.
Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

Ejemplos de modelos iterativos:

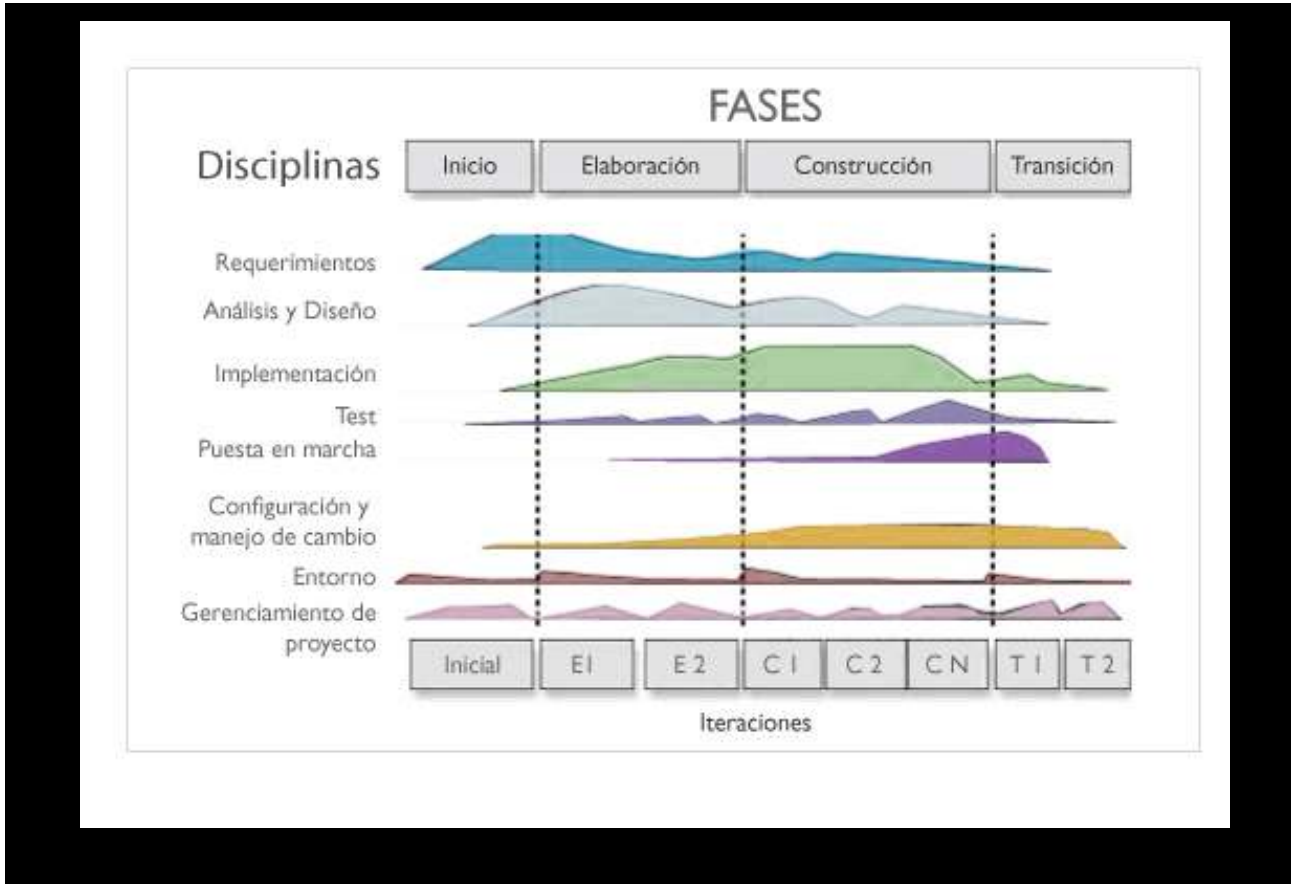
- **Desarrollo rápido de aplicaciones ("Rapid Application Development" - (RAD))**: La interfaz de usuario se implementa utilizando una funcionalidad hecha ("out-of-the-box *) simulando la funcionalidad que posteriormente será desarrollada. Programación extrema



Capitulo 2 - Pruebas a lo largo del ciclo de vida software
1 -Modelos de desarrollo software

Ejemplos de modelos iterativos:

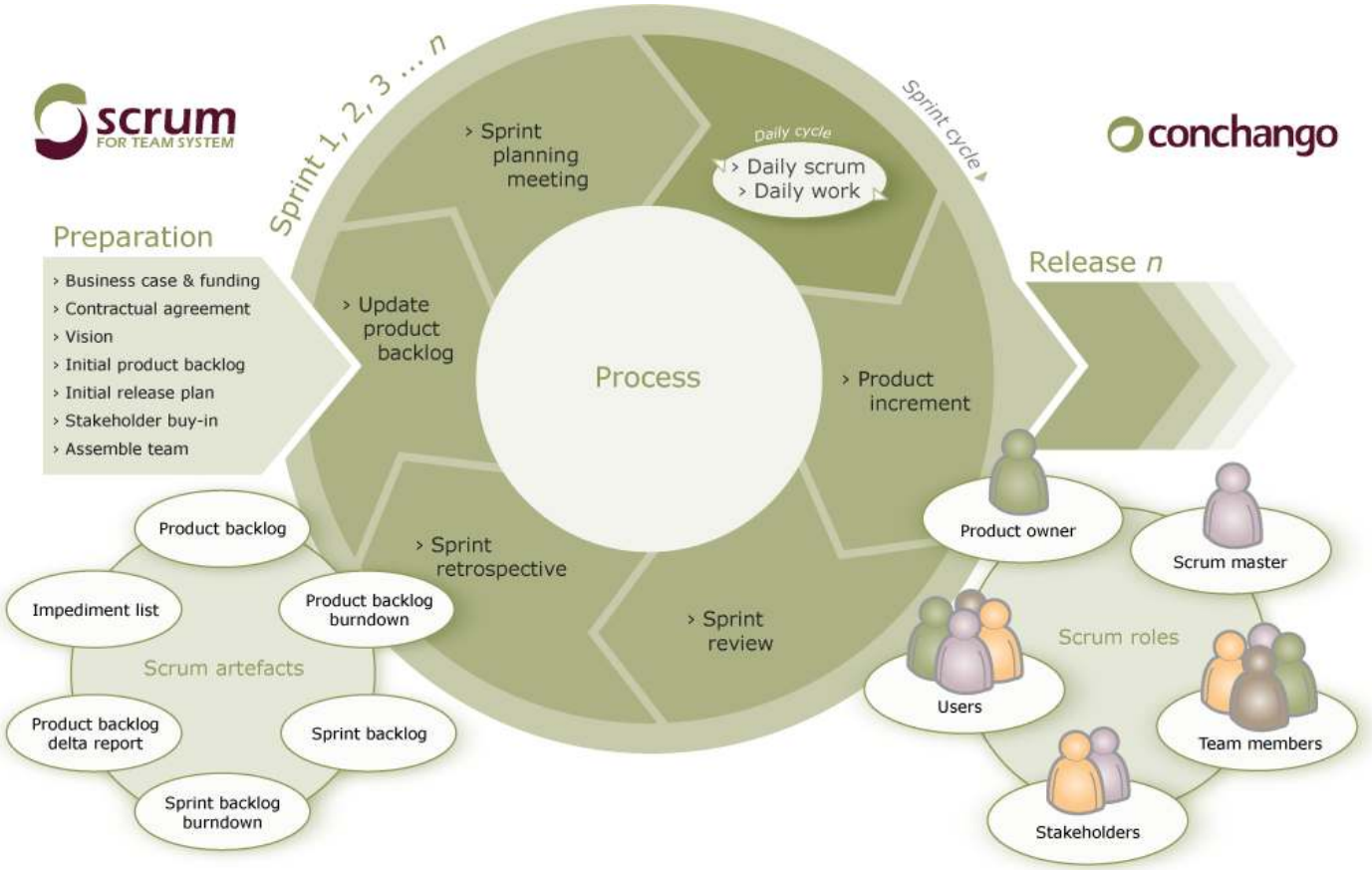
- **Proceso unificado ("Rational Unified Process" - (RUP)):** es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.



Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

Ejemplos de modelos iterativos:

- **SCRUM:** Permite el desarrollo ágil y valora al software que funciona, por encima de la documentación exhaustiva.



2.1.3. Pruebas en un modelo de ciclo de vida.

Para cada modelo de desarrollo de software se deberán aplicar las buenas practicas para unas buenas pruebas.

- Para cada actividad de desarrollo existe una actividad de prueba correspondiente.
- Cada uno de los niveles de prueba deberá tener objetivos de prueba específicos.
- Las actividades de análisis y diseño de las pruebas deberán comenzar durante la ejecución de las actividades de desarrollo.
- Las personas del equipo de pruebas deberán tener acceso a los entregables o a sus borradores de cada uno de los niveles de desarrollo del ciclo de vida, en el momento en que estén disponibles.

Los niveles de prueba no son estáticos, dependiendo de la naturaleza del proyecto pueden variar en su orden o ser combinados, de la misma manera las pruebas se ajustan para que cada nivel de desarrollo tenga su contraparte de pruebas.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.1 -Modelos de desarrollo software

Resumen

- Los modelos de desarrollo de software integran el proceso de pruebas como parte integral del modelo.
- El modelo de desarrollo mas utilizado es el modelo en V, mostrando en paralelo actividades de desarrollo y actividades de pruebas
- Los modelos iterativos también implementan fases de pruebas en cada uno de sus ciclos.
- Cada una de las fases del proceso de desarrollo es susceptible de realizar pruebas validando y verificando sus objetivos.

2.2: Niveles de prueba

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

Términos

- **Pruebas alfa:** Las pruebas operativas simuladas o reales de los usuarios potenciales o clientes un equipo de pruebas independientes en el sitio de los desarrolladores, pero fuera de la organización para el desarrollo.
- **Pruebas beta:** Pruebas de funcionamiento por los usuarios potenciales y / o existentes y clientes en una ubicación externa, no interactúan con los desarrolladores, para determinar si o no un componente o el sistema satisface las necesidades del usuario / cliente y ajustes en los procesos de negocio. Beta.
- **Pruebas de componente:** La prueba de componentes individuales de software [IEEE 610]
- **Controlador (Driver):** Un componente de software o herramienta de pruebas que remplaza a un componente que se encarga de hacer llamadas a otros componentes o sistemas .
- **Campo de pruebas:** Igual a pruebas beta.
- **requerimiento funcional:** Un requisito que especifica una función que un componente o sistema debe llevar a cabo. [IEEE 610].
- **Integración:** El proceso de integración de componentes o sistemas ensamblados.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

Términos

- **Pruebas de integración:** Prueba que se realiza para exponer los defectos en las interfaces y las interacciones entre los componentes integrados o sistemas.
- **Requerimiento no funcional:** El requisito de que no se refiere a la funcionalidad, Para atributos tales como la fiabilidad, eficiencia, facilidad de uso, facilidad de mantenimiento y portabilidad.
- **Pruebas de robustez:** Prueba para determinar la robustez del producto de software.
- **Agente de sustitución (stub):** Una aplicación especial del esqueleto o de propósito de un componente de software, que se utiliza para desarrollar o probar un componente que llama o que sea dependiente de ella. Sustituye a una llamada de un componente. [Después de IEEE 610]
- **Pruebas de sistema:** El proceso de prueba de un sistema integrado para verificar que cumple los requisitos especificados. [Hetzel].
- **Nivel de prueba:** Un grupo de actividades de prueba de que están organizados y gestionados conjuntamente. El nivel de la prueba esta vinculado a los niveles de desarrollo de un proyecto.

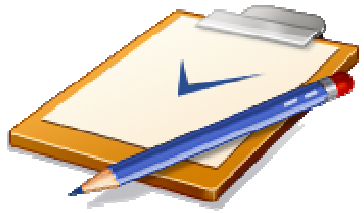
Términos

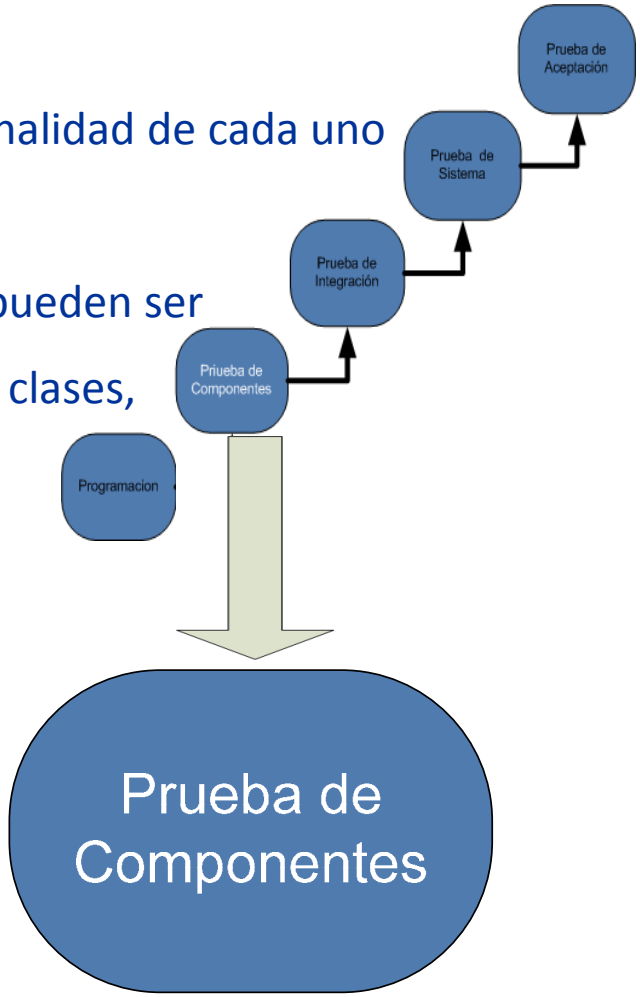
- **Ambiente de prueba:** Un entorno que contiene el hardware, la instrumentación, los simuladores, herramientas de software y otros elementos de apoyo necesarios para llevar a cabo una prueba. [IEEE 610].
- **Pruebas de aceptación de usuario:** Pruebas para asegurarse que el sistema cumple con los requisitos esperados.
- **Desarrollo guiado por prueba:** Una manera de desarrollar software, donde los casos de prueba se desarrollan, y a menudo son automatizados, antes de que el software esté desarrollado para aplicar los casos de prueba.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

2.2.1 Pruebas de componente.

- Usualmente llamadas pruebas unitarias.
- Son desarrolladas para buscar defectos o probar la funcionalidad de cada uno de los elementos de la Aplicación de manera individual .
- Para cada entorno de desarrollo los artefactos de prueba pueden ser Conocidos de forma diferente, como módulos, programas, clases, Objetos, etc.).
- Los desarrolladores ejecutan pruebas unitarias recurrentes durante el desarrollo de los componentes.





Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

Pruebas de componente

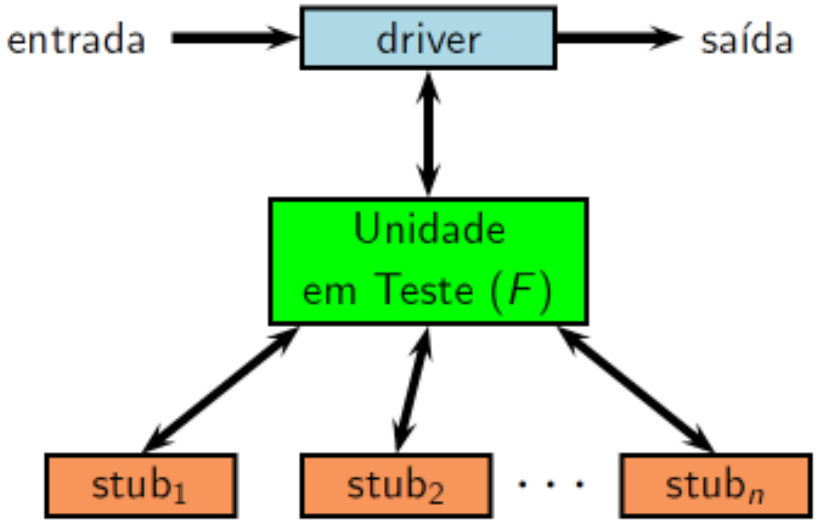
- Las pruebas de componente pueden incluir pruebas de funcionalidad y características no funcionales.
- Cada componente deberá probarse de forma independiente
- Generalmente se tiene acceso al código fuente y los defectos son arreglados tan pronto son encontrados.
- Las pruebas unitarias deben cumplir con algunas características que por la naturaleza de unitarias no se pueden omitir .
  - **Completas:** deberán contemplar la mayor parte de los requerimientos que van a ser atendidos por el artefacto.
  - **Automatizables:** Las pruebas unitarias deben buscar ser automatizadas para poder repetir el proceso de pruebas sobre el componente de una forma iterativa.
  - **Repetibles:** Deberán poder ser utilizadas repetidamente, incluso en pruebas de integración.
  - **Independientes:** No se afectara el resultado de una prueba por la ejecución de otra.
  - **Profesionales:** deberán estar basadas en una metodología y debidamente documentadas.

Pruebas de componente

- Por tratarse de pruebas individuales de componentes que hacen parte de un sistema algunas veces es necesario implementar técnicas que permitan que el funcionamiento del componente no se vea afectado por la ausencia de otros componentes

.

- Para solucionar esto se plantea la utilización de **Drivers** y **Stubs**

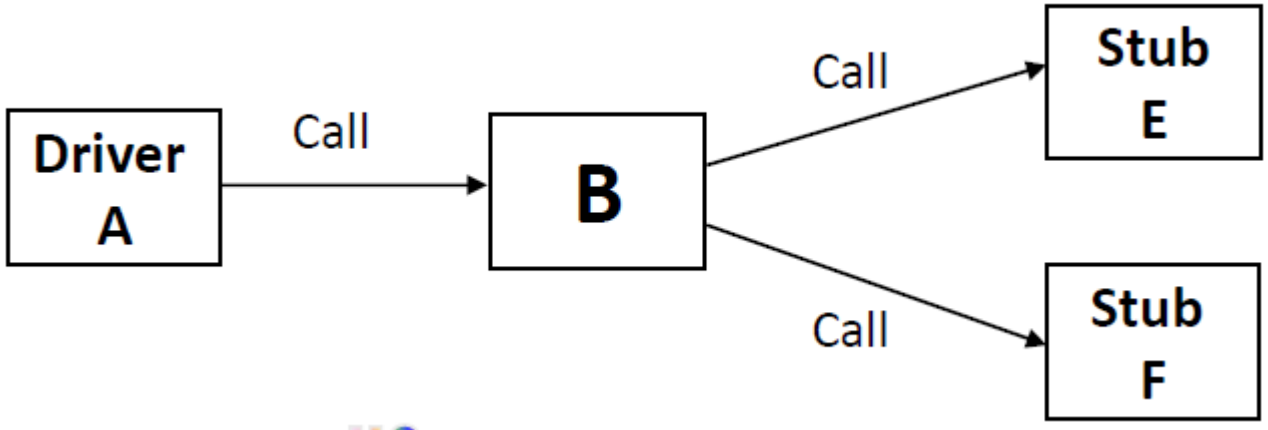


http://www.ironiacorp.com/apps/wiki/testing/Drivers_and_stubs_on_unit_testing

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

Pruebas de componente

Driver: es un componente de código que interactúa con las entradas o salidas requeridas para el funcionamiento del componente que será probado, pueden funcionar como métodos dentro del mismo código objeto de la prueba.



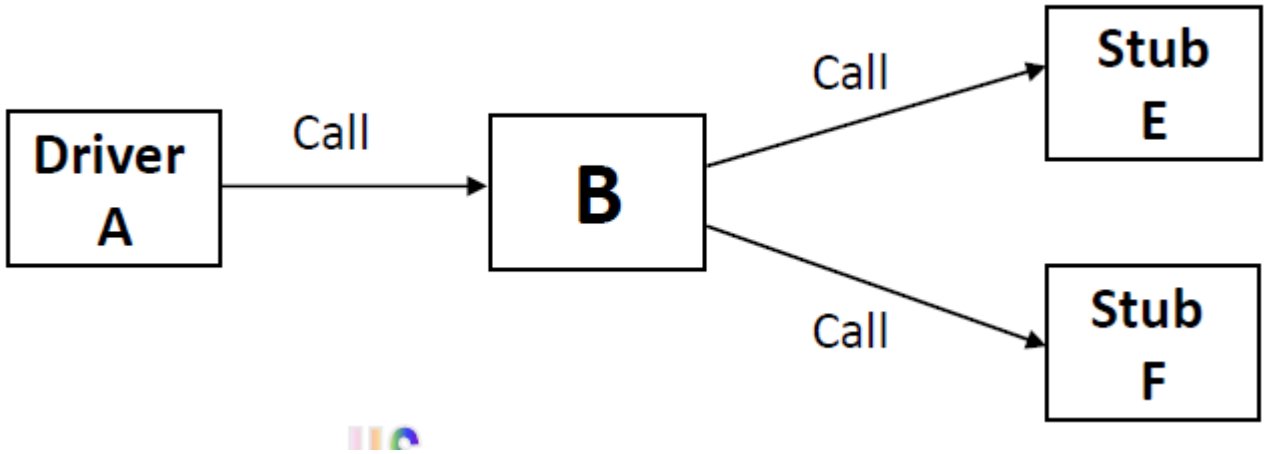
Quiero probar al módulo B de forma aislada – No uso los módulos A, E y F

- El módulo B es usado por el módulo A
  - Debo simular la llamada del modulo A al B – *Driver*
  - Normalmente el Driver es el que suministra los datos de los casos de prueba

http://www.fing.edu.uy/tecnoinf/cursos/ingsoft/material/teorico/IS-07b-Verificacion_Validacion_tecnologo.pdf

Pruebas de componente

Stub: Es un elemento que simula el comportamiento de un elemento que aun no ha sido construido, por ejemplo retornar los valores de una consulta a una tabla, etc.



- El módulo B usa a los módulos E y F
  - Cuando llamo desde B a E o F debo simular la ejecución de estos módulos – *Stub*

http://www.fing.edu.uy/tecnoinf/cursos/ingsoft/material/teorico/IS-07b-Verificacion_Validacion_tecnologo.pdf

Pruebas de componente: métodos de prueba

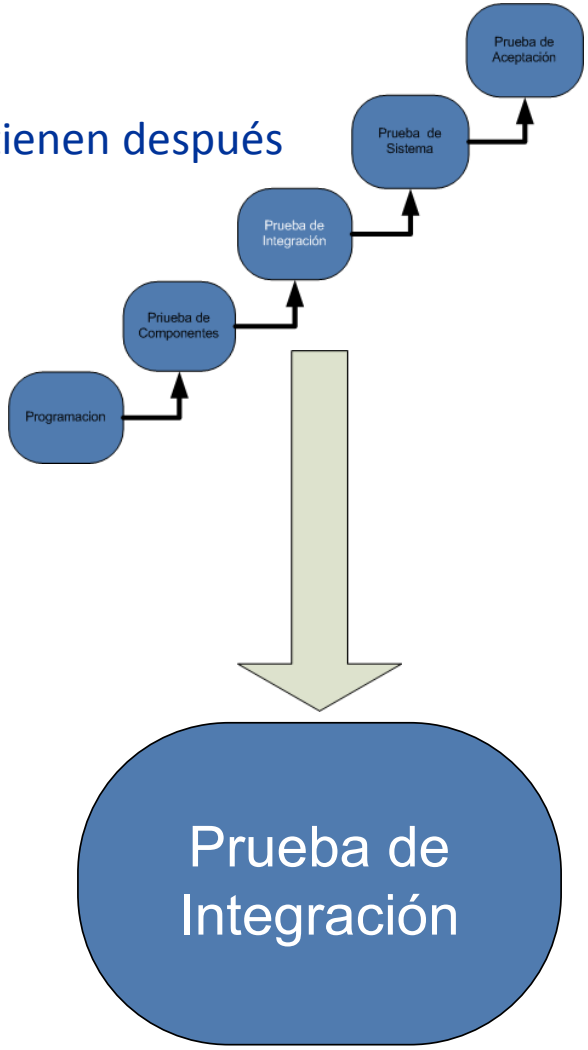
- El código fuente se encuentra disponible para el probador (" tester").
  - Caso probador ('tester') = desarrollador: Las pruebas se desarrollan con acceso y soporte en ambiente de desarrollo.
  - Se podrá aplicar al diseño de casos de prueba el conocimiento de la funcionalidad, estructura de componentes y variables.
  - Las pruebas funcionales serán periódicas.
  - Adicionalmente, el uso de depuradores ("debuggers") y otras herramientas de desarrollo permitirán acceso directo a las variables del programa.
  - Las técnicas de caja blanca (“WhiteBox”) pueden ser utilizadas debido al acceso al código fuente del componente.
  - Se podrán probar características no funcionales como la robustez, pruebas de estrés, rendimiento etc.

- **Resumen: pruebas de componente**

- Un componente es la unidad más pequeña especificada de un sistema.
- Prueba de módulo, de clase, de unidad y de desarrollador son utilizados como sinónimos.
- Los "drivers" ejecutarán las funciones de un componente y las funciones adyacentes que son remplazadas por "stubs"
- Las pruebas de componente podrán comprobar características funcionales y no funcionales de un sistema.

2.2.2. Pruebas de integración

- Es realizada posterior a las pruebas unitarias, se desarrolla sobre todos los componentes del sistema de una sola vez.
- Permite validar que las funcionalidades del sistema se mantienen después de integrar los componentes del mismo.
- A diferencia de las pruebas unitarias no se centra en la funcionalidad de cada uno de los módulos sino como un grupo o conjunto.
- Las pruebas de integración preceden Las pruebas del sistema.
  Las pruebas de integración pueden incluir pruebas de características funcionales y no funcionales

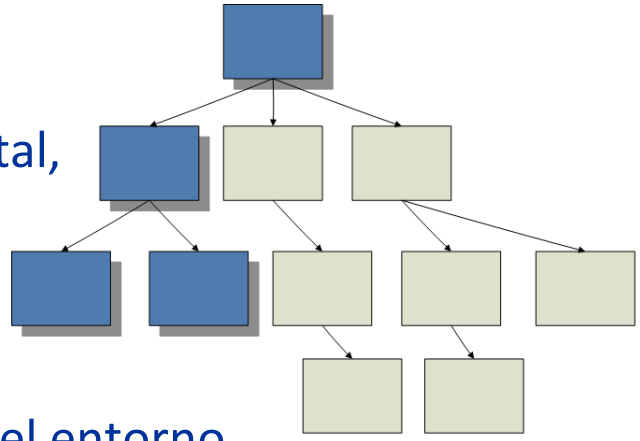


Pruebas de integración

- Las pruebas de integración pueden contemplar la totalidad del sistema o algunas de sus partes.

Pruebas de integración: De componentes

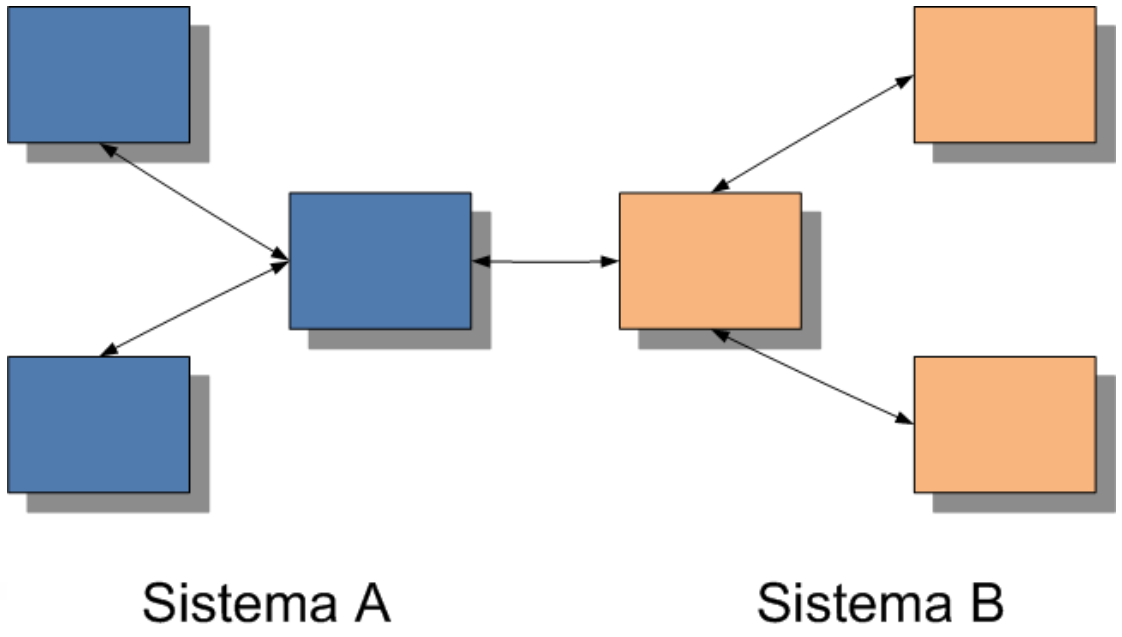
- Busca probar la integración de una parte de los componentes del sistema y sus iteraciones entre si.
- La integración a menudo se hace de forma incremental, diferente del enfoque BigBang (Todo al tiempo)
- Se buscan defectos asociados a la interfaz, tanto Interna (otros módulos) como con otros elementos del entorno.
- Pueden ser necesarios Drivers o Stubs para simular los elementos que no están incluidos en la integración.



Pruebas de integración

Pruebas de integración: De sistema

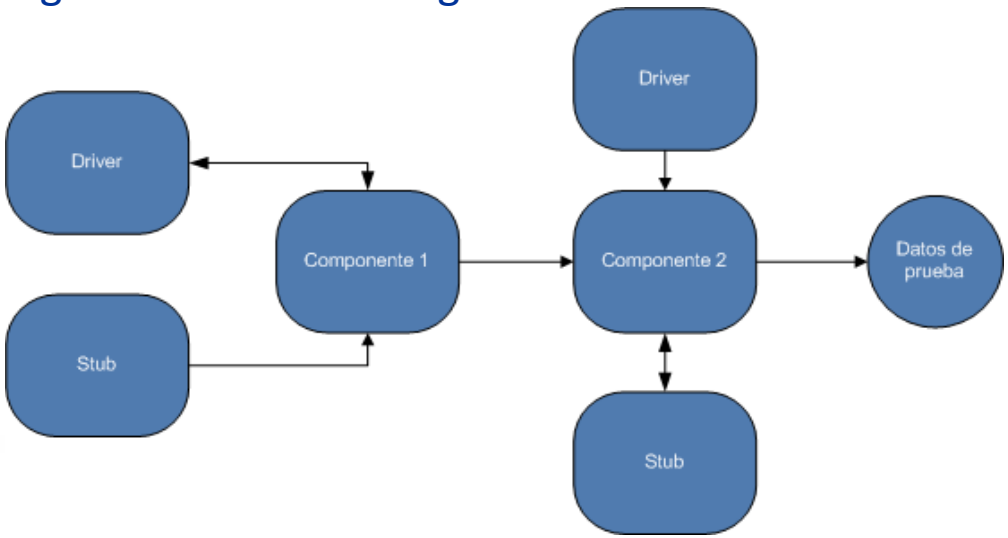
- La integración se realiza con otros sistemas
- Los defectos hallados pueden ser desestabilizadores y significativos
- Entre mas sistemas estén integrados, mas difícil será la individualización del fallo.



Pruebas de integración

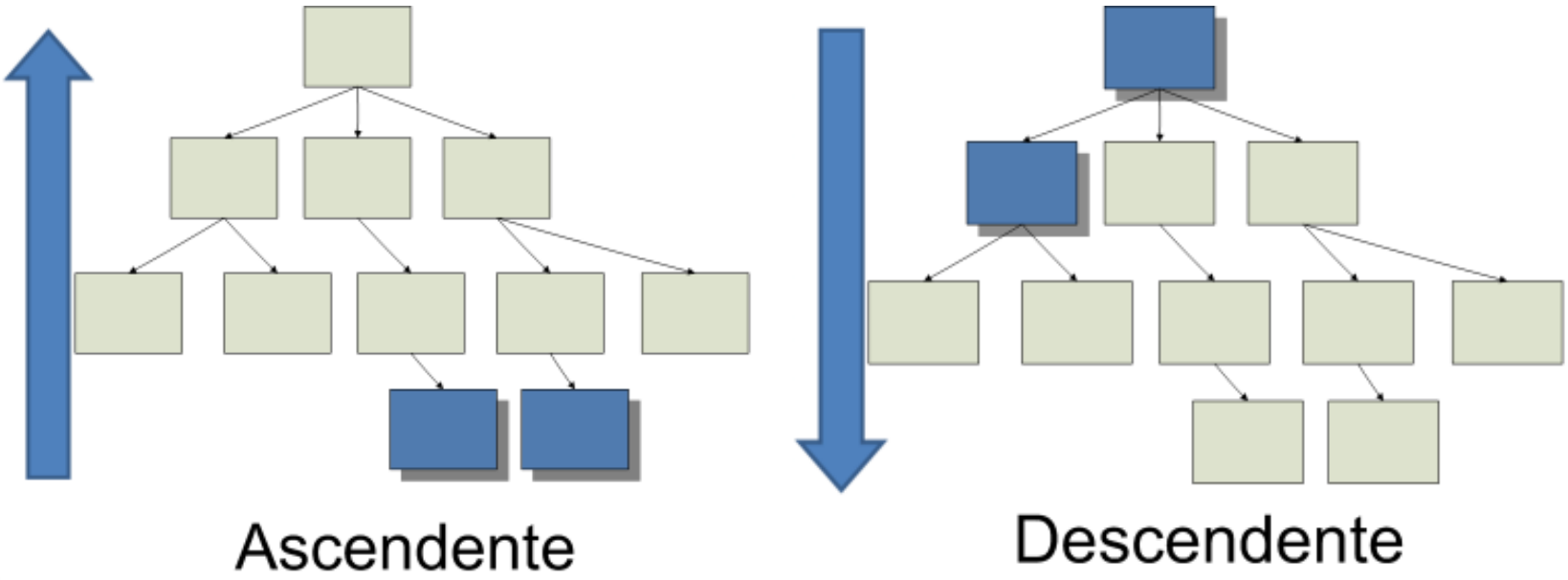
Stubs y Drivers

- Los "drivers" de prueba propios de las pruebas de componente pueden ser **reutilizadas** aquí
- Un “Stub" igual que en las pruebas de componente remplaza un elemento faltante.
  - Un “Stub" programado remplazará datos o funcionalidad de un componente que aún no ha sido integrado.
  - Un “Stub" asumirá las tareas elementales de un componente faltante.
- Al remplazar los “Stub” o Drivers por los elementos faltantes se pueden encontrar nuevos defectos generados de la integración



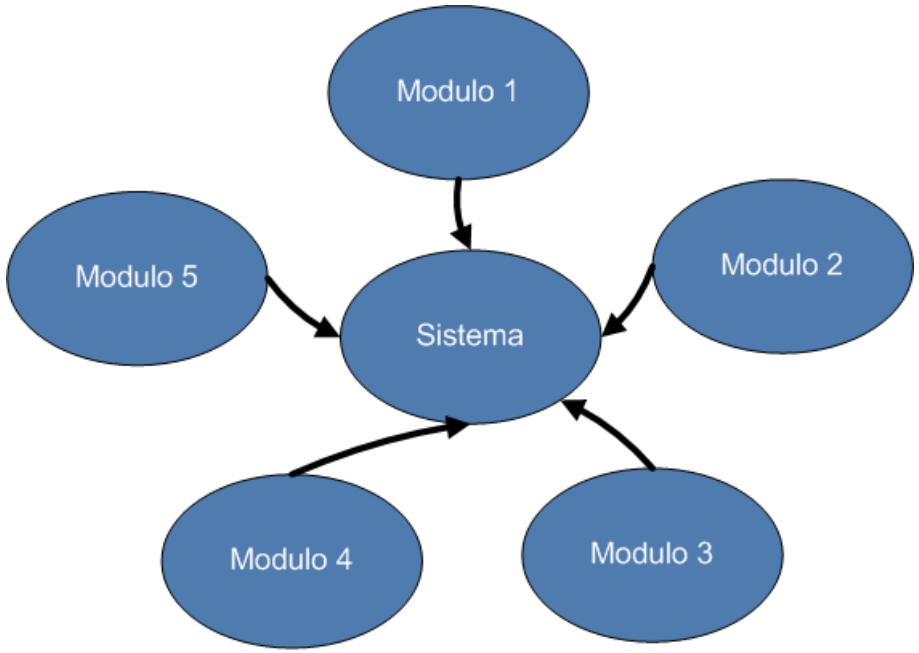
Pruebas de integración: Estrategias

- El **enfoque incremental** se prueba a medida que los componentes están disponibles incluyendo los componentes anteriores.
- Las estrategias **ascendente** ("bottom-up") y **descendente** ("top-down") son las utilizados con mayor frecuencia.



Pruebas de integración: Estrategias

- Estrategia Big Bang, se espera que todos los componentes del sistema estén terminados para probar la integración.
- Se prueba cada módulo de forma aislada y luego se prueba la combinación de todos los módulos a la vez.



Big Bang

Pruebas de integración: Estrategias

- De acuerdo a las características de cada sistema se deberá elegir una estrategia que brinde los mejores resultados.
- Las estrategias que no están mencionadas normalmente se conocen como estrategias “Ad Hoc”

Resumen: pruebas de integración

- La integración se refiere a la interacción entre componentes o sistemas
- Las pruebas de integración validan el comportamiento de las interfaces de los componentes
- La estrategia puede ser Ascendente, Descendente, BigBang o Ad Hoc
- También son consideradas pruebas de integración las que se realizan entre sistemas comunicados entre si.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.2 - Niveles de prueba

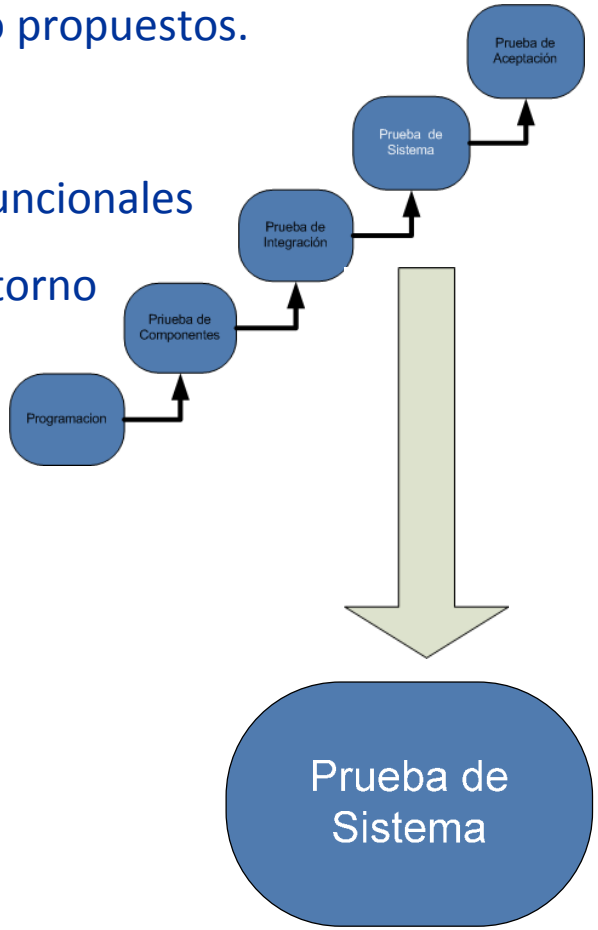
2.2.3. Pruebas de sistema

- Las pruebas de sistema buscan comprobar el software de manera global validando y verificando que cumpla con los requerimientos solicitados o propuestos.
- Usualmente las pruebas de sistema son de caja negra
- Las pruebas de sistema incluyen pruebas funcionales y no funcionales
- En este tipo de pruebas se pueden incluir elementos del entorno

  Como personas, hardware, etc.
- Es recomendable que se involucren los usuarios en las

  pruebas de sistema debido a que son los que tienen

  mas conocimiento del negocio.



Pruebas de sistema: Características

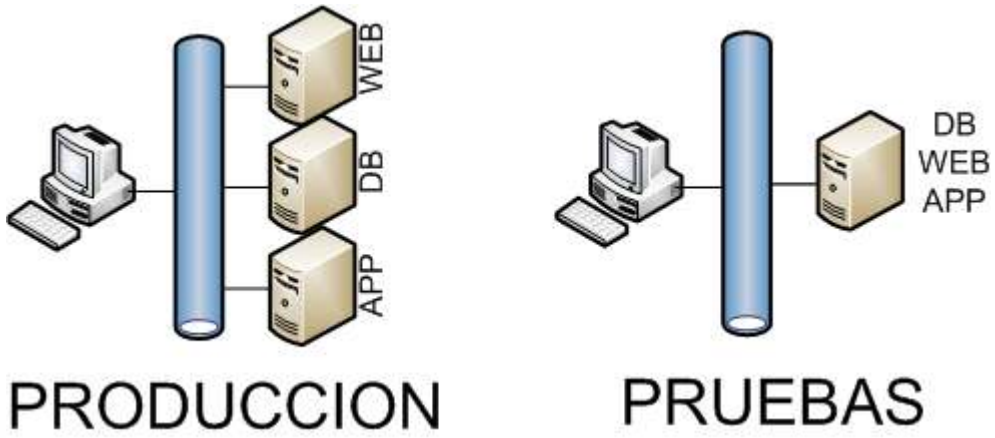
- Las pruebas de sistema deberán llevarse a cabo sobre ambientes similares al ambiente definitivo para la aplicación

  "No se realizaran pruebas en ambientes reales o de producción"

- Si se han desarrollado las pruebas de las fases anteriores (De componente y de Integración) se reduce el numero de defectos que se pueden hallar en esta fase del ciclo de vida del proyecto.

- Ya no son necesarios los "Stub" o "Drivers" , el sistema debería estar completo.

- Se pueden utilizar ambientes pre-productivos para emular un comportamiento lo mas real posible.



Pruebas de sistema: Características

Comprobar que la funcionalidad atiende los requerimientos funcionales y no funcionales propuestos

Las características pueden ser (según norma ISO 9126)

Funcionales	No Funcionales
• Adecuación/idoneidad ("suitability") • Precisión (“accuracy") • Conformidad ("compliance") • Interoperabilidad ("interoperability“) • Seguridad ("security").	• Fiabilidad • Usabilidad • Mantenibilidad • Portabilidad

Pruebas de sistema: Enfoques

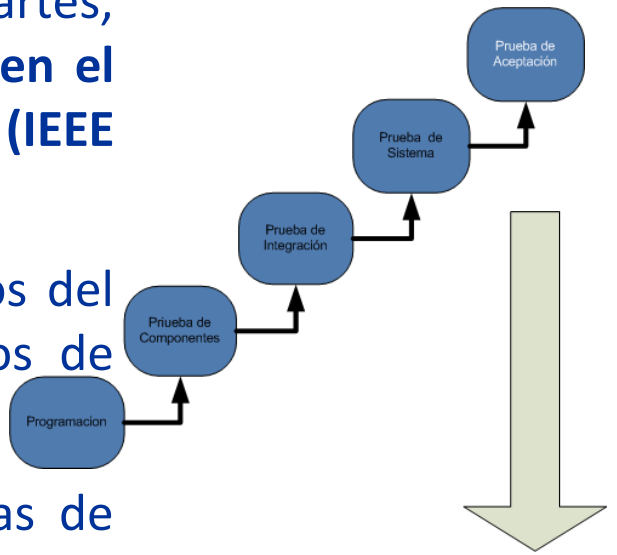
- Pruebas basadas en requisitos y/o especificaciones de requerimientos.
- Pruebas basadas en procesos de negocio.
- Pruebas basadas en casos de uso.
- Pruebas basadas en reglas y/o procesos de negocio
- Pruebas basadas en comportamientos de alto nivel
- Pruebas basadas en interacciones con otros sistemas

Resumen: pruebas de sistema

- Las pruebas de sistema se desarrollan utilizando casos de prueba funcionales y no funcionales, apoyados en herramientas de creación de casos de prueba de caja negra y caja blanca
- Las pruebas funcionales del sistema confirman que los requisitos para un uso específico previsto han sido satisfechos (validación).
- Las pruebas de sistema no funcionales verifican los atributos de calidad no funcionales, por ejemplo usabilidad, eficiencia, portabilidad, etc.
- Con frecuencia, los atributos de calidad no funcionales son una parte implícita de los requisitos, esto hace difícil validarlos

2.2.4 Pruebas de Aceptación: Definición

- Busca establecer confianza en el sistema o en sus partes, **el objetivo es aportar justificación a la confianza en el sistema para que pueda ser aceptado por el cliente. (IEEE 610)**
- Son a menudo responsabilidad del cliente o usuarios del sistema, sin embargo se pueden involucrar equipos de pruebas externos.
- Encontrar defectos no es el objetivo de las pruebas de aceptación
- Se utilizan también para validar si un producto/software es el adecuado para la organización
- Las pruebas de aceptación pueden ser desarrolladas en cualquiera de los niveles de prueba

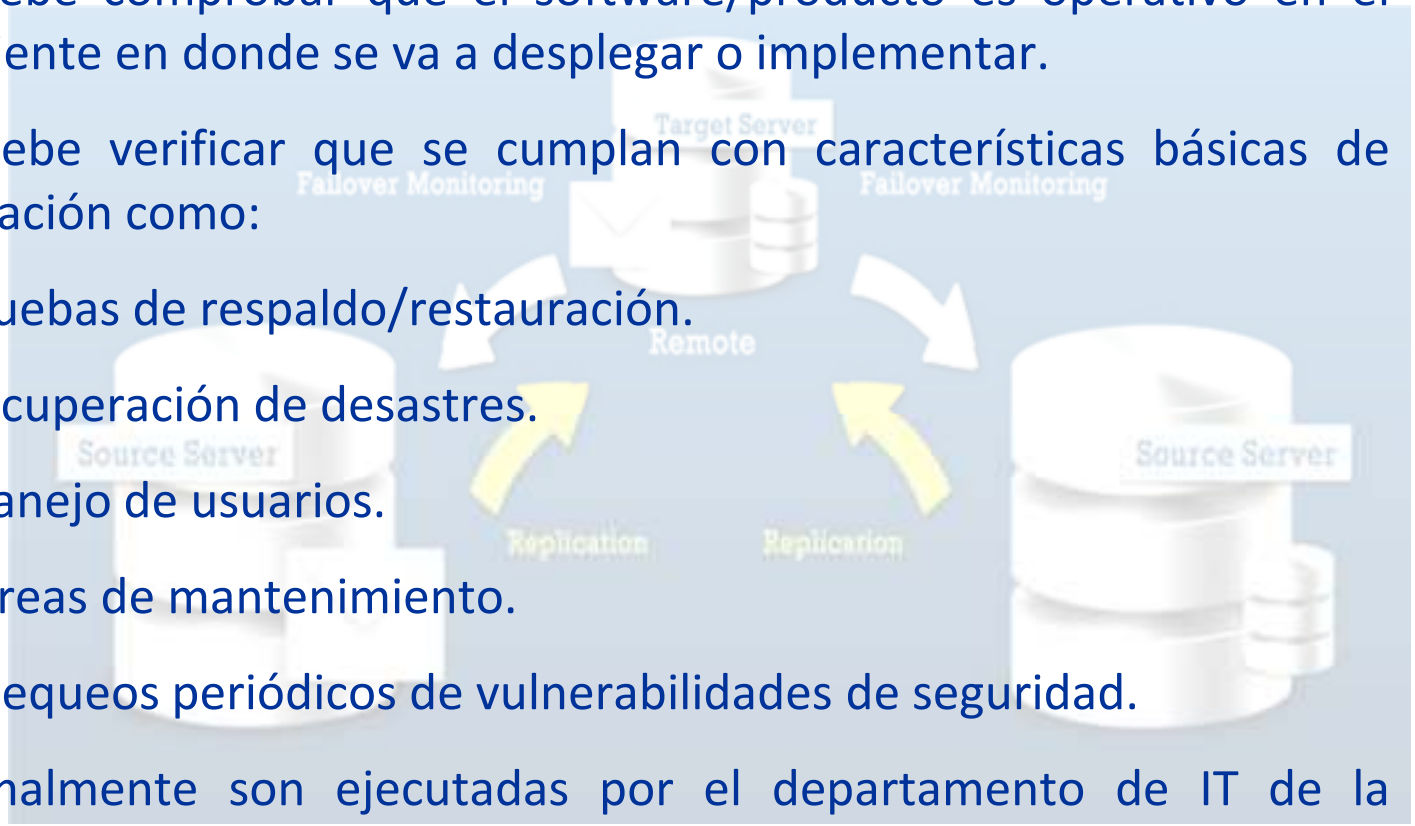


Pruebas de aceptación: Contrato y regulaciones

- ¿El software satisface todos los requisitos de normativas vigentes?
  - Las pruebas de aceptación deben tener en cuenta normas y reglamentos gubernamentales, legales, industriales y de otro tipo (por ejemplo reglamento de manejo de crédito de la Superintendencia Bancaria)
  - Las pruebas de aceptación deberán validar que los compromisos contractuales se cumplan a cabalidad, atendiendo todos los requerimientos solicitados.
  - Se cumplen los parámetros para la aceptación de garantías de ambas partes.
- Los casos de prueba son seleccionados por el cliente y/o usuario, es quien conoce el negocio y puede mitigar el riesgo de malas interpretaciones.
  - Las pruebas se realizan utilizando el entorno del cliente.
- EL cambio de entornos puede hacer evidentes defectos que no se detectaron en ambientes de pruebas.

Pruebas de aceptación: Pruebas Operacionales

- Se debe comprobar que el software/producto es operativo en el ambiente en donde se va a desplegar o implementar.
- Se debe verificar que se cumplan con características básicas de operación como:
  - Pruebas de respaldo/restauración.
  - Recuperación de desastres.
  - Manejo de usuarios.
  - Tareas de mantenimiento.
  - Chequeos periódicos de vulnerabilidades de seguridad.
- Normalmente son ejecutadas por el departamento de IT de la organización



Pruebas de aceptación: Pruebas Alpha Beta (de campo)

- Son las realizadas a nivel de desarrolladores quienes pretenden obtener retroalimentación por parte de clientes existentes o potenciales.
- Se entregan versiones estables de aplicativos completamente funcionales.
- Normalmente son utilizados para actividades cotidianas, ocasionalmente se desarrollan casos de pruebas para su verificación.
- Las pruebas Alpha y Beta incluyen la prueba en variedad de ambientes por cada uno de los clientes que lleven a cabo la prueba.
- Las pruebas Alpha son desarrolladas en las instalaciones del desarrollador y las Beta en las instalaciones del cliente.
- Las pruebas Beta tienen una ventaja y es que se reducen los costos de pruebas para la organización pero se pierde el control de la trazabilidad de los defectos encontrados.

Resumen: pruebas de aceptación

- Las pruebas de aceptación son las pruebas de sistema por parte del **cliente.**
- La prueba de aceptación es una actividad de carácter **contractual**, se verificará entonces que el software satisface los requisitos del cliente.
- Las pruebas Operacionales tienen como fin validar el funcionamiento en el ambiente o infraestructura del cliente.
- Las pruebas **alpha y beta** son pruebas ejecutadas por clientes reales o potenciales, en las dependencias del desarrollador (alpha) o en las dependencias del cliente (beta).

2.3: Tipos de pruebas

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Términos

- **Pruebas de caja negra:** El procedimiento para obtener y/o seleccionar casos de prueba basados en el análisis de la especificación, ya sea funcional o no funcional, de un componente o sistema sin referencia a su estructura interna.
- **Cobertura de código:** Un método de análisis que determina qué partes del software han sido ejecutados (cubiertos) por el set de pruebas y qué partes no han sido ejecutados.
- **Pruebas funcionales:** Pruebas basadas en el análisis de la especificación de una funcionalidad o de un componente o de un sistema.
- **Pruebas de interoperabilidad:** El proceso de probar para determinar la interoperabilidad de un producto de software .
- **Pruebas de carga:** Un tipo de pruebas de rendimiento a cabo para evaluar el comportamiento de un componente o sistema con el aumento de carga, por ejemplo número de usuarios en paralelo y / o números de transacciones, para determinar qué carga puede ser manejado por el componente o sistema

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Términos

- **Pruebas de rendimiento:** El proceso de probar para determinar el rendimiento de un producto de software .
- **Pruebas de portabilidad: :** El proceso de probar para determinar la portabilidad de un producto de software .
- **Pruebas de confiabilidad:** El proceso de probar para determinar la confiabilidad(fiabilidad) de un producto de software .
- **Pruebas de seguridad:** El proceso de probar para determinar la seguridad de un producto de software
- **Pruebas basadas en especificación:** Ver también pruebas de caja negra.
- **Pruebas de estrés:** Un tipo de pruebas de rendimiento realizadas para evaluar un sistema o componente en o más allá de los límites de limite de trabajo previsto o especificado, o con menor disponibilidad de los recursos como el acceso a la memoria o servidores. [IEEE 610]
- **Pruebas de mantenibilidad:** El proceso de probar para determinar la mantenibilidad de un producto de software .

Términos

- **Pruebas estructurales:** Ver pruebas de caja blanca
- **Pruebas de usabilidad:** Las pruebas para determinar el grado en el que el producto de software es fácil de aprender, fácil de operar y atractivo para los usuarios en determinadas condiciones. [norma ISO 9126]
- **pruebas de caja blanca:** Pruebas basadas en el análisis de la estructura interna de un componente o sistema.

Tipos de pruebas por nivel

En cada uno de los niveles de pruebas explicados en la sección anterior se definieron los objetivos de las pruebas, por consiguiente para cada nivel de pruebas existe uno o varios tipos de prueba que aplican para validar el cumplimiento de los requerimientos del nivel.

Tipos de pruebas.

- Pruebas funcionales (Objetivo: probar la función).
- Pruebas no funcionales (Objetivo: probar las características del producto),
- Pruebas estructurales (Objetivo: probar la estructura/arquitectura software).
- Pruebas de confirmación/regresión (Objetivo: probar después de cambios).

2.3.1. Pruebas de función

- Son las que validan y verifican cada una de los requerimientos funcionales plasmados en las especificaciones del software/producto, teniendo en cuenta sus entradas y salidas basándose en un modelo de caja negra.
- Generalmente son desarrolladas por equipos de pruebas conformados por analistas de pruebas y usuarios finales como apoyo.
- Las pruebas funcionales se pueden llevar a cabo en todos los niveles de prueba.
- El objeto de prueba es ejecutado utilizando combinaciones de datos de prueba derivados/generados a partir de los casos de prueba.
- Los resultados de la ejecución de la prueba son comparados con los resultados esperados.
- Cuando se han desarrollado las pruebas unitarias las pruebas funcionales requieren menos tiempo de ejecución.

2.3.2. Pruebas de características de software no funcionales

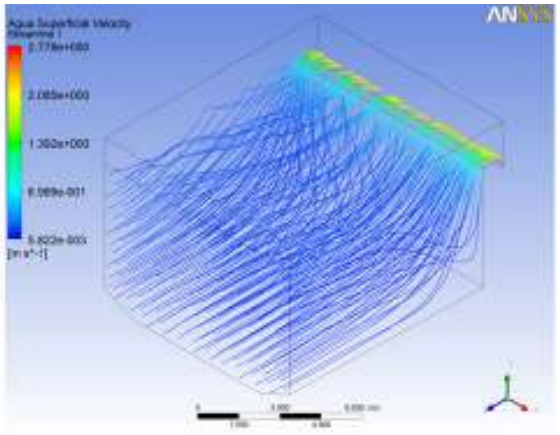
Busca verificar que el sistema se comporte “como” se supone que debe comportarse.

- Las pruebas no funcionales pueden llevarse a cabo en cualquiera de los niveles de pruebas.
- Pruebas no funcionales típicas.
  - Pruebas de desempeño
  - Pruebas de carga
  - Pruebas de estrés
  - Pruebas de usabilidad
  - Pruebas de mantenibilidad
  - Pruebas de confiabilidad
  - Pruebas de portabilidad

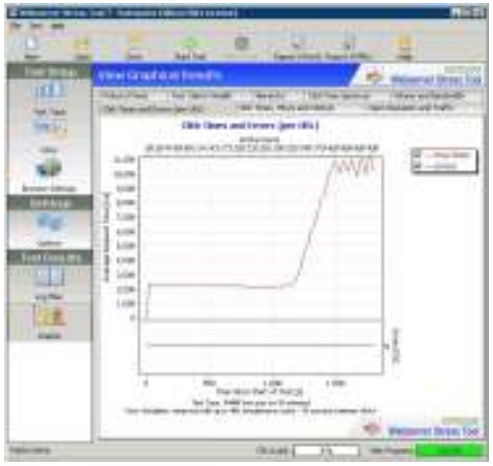
Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Pruebas no funcionales (pruebas de sistema)

- **Pruebas de desempeño :** son las utilizadas para medir las características del sistema que pueden ser medibles como los tiempos de respuesta o la utilización de los recursos de memoria en la infraestructura.



http://puntoedu.pucp.edu.pe/noticias/nuevo-software-para-simulaciones-en-ingenieria-mecanica/



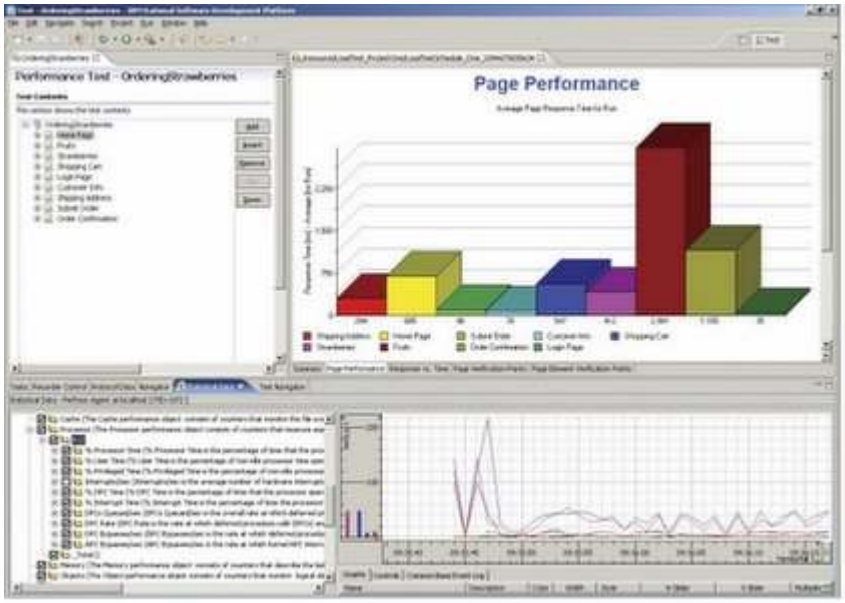
http://www.microteching.com/servidores/pruebas-de-carga-stress-de-un-servidor-web

- **Pruebas de Carga:** Son las que buscan validar que el sistema soporte la cantidad de usuarios, peticiones, llamados, etc., que se han definido en el plan de pruebas.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Pruebas no funcionales (pruebas de sistema)

- **Pruebas de estrés** : Son las orientadas a comprobar la resistencia del sistema y el comportamiento del mismo frente a los posibles fallos. **Reacción a la sobrecarga/recuperación tras el retorno de una carga normal.**



http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246

http://yjlizarazo.wordpress.com/2010/09/

- **Pruebas de Confiabilidad:** Son las encargadas de validar que el sistema incluya controles de acceso y no tenga huecos de seguridad que afecten a los demás sistemas y a la información.

Pruebas no funcionales (pruebas de sistema)

- **Pruebas de usabilidad** : Busca comprobar la facilidad con la que el usuario final aprende y utiliza la aplicación, esta basada en buenas practicas de programación.
- **Pruebas de mantenibilidad**: Buscan comprobar la calidad del software frente a cambios en su código fuente y en la complejidad para afrontar dichos cambios.
- **Pruebas de portabilidad**: Comprueba la facilidad con la que un software es migrado hacia un entorno diferente, valida la facilidad de instalación, ajuste, etc.

2.3.3. Pruebas de estructura/arquitectura de software (pruebas estructurales)

El objetivo de las prueba estructurales es validar la cobertura del software/producto

- La finalidad de las pruebas es medir el grado en el cual la estructura del objeto de prueba ha sido cubierto por los casos de prueba.
- Las pruebas estructurales son posibles en todos los niveles de pruebas, se realizan de forma conjunta a las pruebas de componente y de integración mediante el uso de herramientas.
- El diseño de pruebas estructurales se finaliza tras haber sido diseñadas las pruebas funcionales, con el propósito de obtener un alto grado de cobertura.
- Se deberán probar todas las posibles combinaciones de elementos de un sistema, se mencionan las técnicas de cobertura mas adelante.
- Las pruebas estructurales también pueden ser utilizadas para verificar los caminos de decisión dentro de la aplicación.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Pruebas relacionadas a cambios (pruebas de confirmación (Retest) y pruebas de regresión)

- El “Retest” busca verificar que los defectos se hayan solucionado por medio de una nueva ejecución.
- La “Regresión” busca verificar que no surgieron mas defectos con la solución de los defectos que ya habían sido detectados.
- El alcance de las pruebas de regresión depende del riesgo que la nueva funcionalidad implementado (extensión *o* corrección de errores) impone al sistema.
- Las pruebas de confirmación /regresión pueden ser realizados en todos los niveles.
- Se dividen en:

  Confirmación (Retest): Es la ejecución de los casos de prueba que generaron errores para confirmar la solución del mismo.

  Regresión: Es la ejecución de algunos de los casos de prueba para confirmar que no se han modificado las funcionalidades que funcionaban bien y que no se han generado mas defectos con la solución de los anteriores.

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Regresión y Confirmación

- Se pueden utilizar los casos de prueba utilizados en las pruebas anteriores reduciendo el tiempo de elaboración de los casos de prueba.
- Las pruebas de regresión o de confirmación no deberán tener el mismo numero de casos de prueba que la prueba funcional porque incrementa los costos de la prueba comprometiendo su viabilidad.
- Pautas para la selección de casos de prueba para regresión

  Casos de prueba críticos en el sistema.

  Casos de prueba que generaron errores en la ejecución.

  Casos de prueba que cubran el camino feliz

  Casos de prueba adyacentes a los que presentaron errores

  Casos de prueba específicos para la organización.

Regresión y Confirmación

- Se valida que no se hayan “Introducido” defectos con las modificaciones o aplicación de controles de cambios
- Las pruebas de regresión aplican para las pruebas Funcionales, no funcionales, estructurales y en todos los niveles.
- De acuerdo al grado de madurez del software se pueden automatizar las pruebas de regresión

Capitulo 2 - Pruebas a lo largo del ciclo de vida software
Tema 2.3 – Tipos de pruebas

Resumen

- Para cada nivel de pruebas se puede utilizar un tipo de prueba diferente.
- Los tipos de pruebas son: **funcionales, no funcionales, estructurales y pruebas relacionadas a cambios**.
- Las **pruebas funcionales** verifican que la aplicación haga lo “Que” debe hacer
- Las **pruebas no funcionales** verifica que la aplicación haga lo que tiene que hacer “Como” lo debe hacer
- Las pruebas no funcionales incluyen, pero no están limitadas a **pruebas de carga, pruebas de estrés, pruebas de rendimiento, pruebas de robustez.**
- Al finalizar las pruebas se hacen necesarias las pruebas de confirmación y regresión, para comprobar que no se hayan generado defectos con los cambios.

2.4: Pruebas de mantenimiento

Términos

- **Análisis de impacto**: Es la evaluación de los cambios en las capas de la documentación de desarrollo, prueba de documentación y componentes, con el fin de implementar cambios específicos a determinados requisitos.
- **Pruebas de mantenimiento:** Prueba de los cambios en un sistema operativo o el impacto de un cambió el entorno de un sistema totalmente operativo.

Pruebas de mantenimiento

- Son aquellas pruebas que se aplican a productos de software que ya han pasado todo su ciclo de desarrollo y están implementados y funcionando en producción y son objeto de modificaciones , actualizaciones, migraciones, etc.
- De acuerdo al tipo de modificación que se realice en el software se pueden plantear diferentes tipos de pruebas.
- Una prueba de mantenimiento se puede basar en las pruebas de regresión aplicadas al sistema antes de su puesta en marcha.
- Se pueden incluir las pruebas de migración de datos como parte de las pruebas de mantenimiento si los cambios aplicados lo ameritan
- Se deberán contemplar pruebas de regresión extensiva a partes del sistema que no han sido modificadas.