Verificación y validación de software

Fundamentos de la verificación y validación del software

Fundamentos de la verificación y validación del software

En la clase anterior…

- Conceptos de Verificación y Validación
- Aseguramiento de la calidad del software
- Costos de los defectos
- Defectos → Mala calidad
- Técnicas dinámicas de verificación
- Pruebas estadísticas

Fundamentos de la verificación y validación del software

Pressman:

- **Verificación** se refiere al conjunto de tareas que garantizan que el software implementa correctamente una función específica.
- **Validación** es un conjunto diferente de tareas que aseguran que el software que se construye sigue los requerimientos del cliente.

- Boehm:

Verificación: ¿Estamos construyendo el producto correctamente?

Validación: ¿Estamos construyendo el producto correcto?

Fundamentos de la verificación y validación del software

Sommerville

Verificación

- Busca comprobar que el sistema cumple con los requerimientos especificados (funcionales y no funcionales)
- ¿El software está de acuerdo con su especificación?

Validación

- Busca comprobar que el software hace lo que el usuario espera.
- ¿El software cumple las expectativas del cliente?

Fundamentos de la verificación y validación del software

IEEE

Verificación

- The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase

Validación

- The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements

Fundamentos de la verificación y validación del software

Las pruebas de software tienen un rol muy importante en el **aseguramiento de la calidad,** ya que permiten detectar los errores introducidos en las fases previas del proyecto

Fundamentos de la verificación y validación del software

- **Costos de la calidad**
  - El costo promedio por corregir un defecto durante la **codificación** es aproximadamente de **US$977** por error.
  - El promedio del costo por corregir el mismo error si se descubre durante las **pruebas** del sistema es de **US$7, 136.**
  - **Según Cigital: una aplicación grande contiene 200 errores**

(200 × US$977) = US$195 400

Pressman 2002

Fundamentos de la verificación y validación del software

Sobredosis radiológica en el Instituto Nacional del Cáncer de Panama (2000)

Errores en los procedimientos y un fallo de software causan que se apliquen dosis erróneas de radiación 8 personas murieron y 20 tuvieron problemas de salud graves.

Los médicos responsables del hecho fueron acusados de asesinato

Fundamentos de la verificación y validación del software

CALIDAD

Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan.

Fundamentos de la verificación y validación del software

- **Aseguramiento de la calidad del software**
  - **Elementos**:
    - **Estándares**.
    - **Revisiones y auditorías**
    - **Pruebas**.
    - Colección y análisis de los errores.
    - Administración del cambio.
    - Educación.
    - Administración de los proveedores.
    - Administración de la seguridad.
    - Seguridad.
    - Administración de riesgos

Fundamentos de la verificación y validación del software

- **Aseguramiento de la calidad del software**
  - **El plan ACS incluye:**
    - Las evaluaciones que se van a realizar
    - Las auditorías y revisiones por efectuar
    - Los estándares aplicables al proyecto
    - Los procedimientos para reportar y dar seguimiento a los errores
    - Los productos del trabajo que genera el grupo de ACS y la retroalimentación que se dará al equipo del software.

Fundamentos de la verificación y validación del software

En el proceso de Verificación y Validación se utilizan dos técnicas de comprobación y análisis:

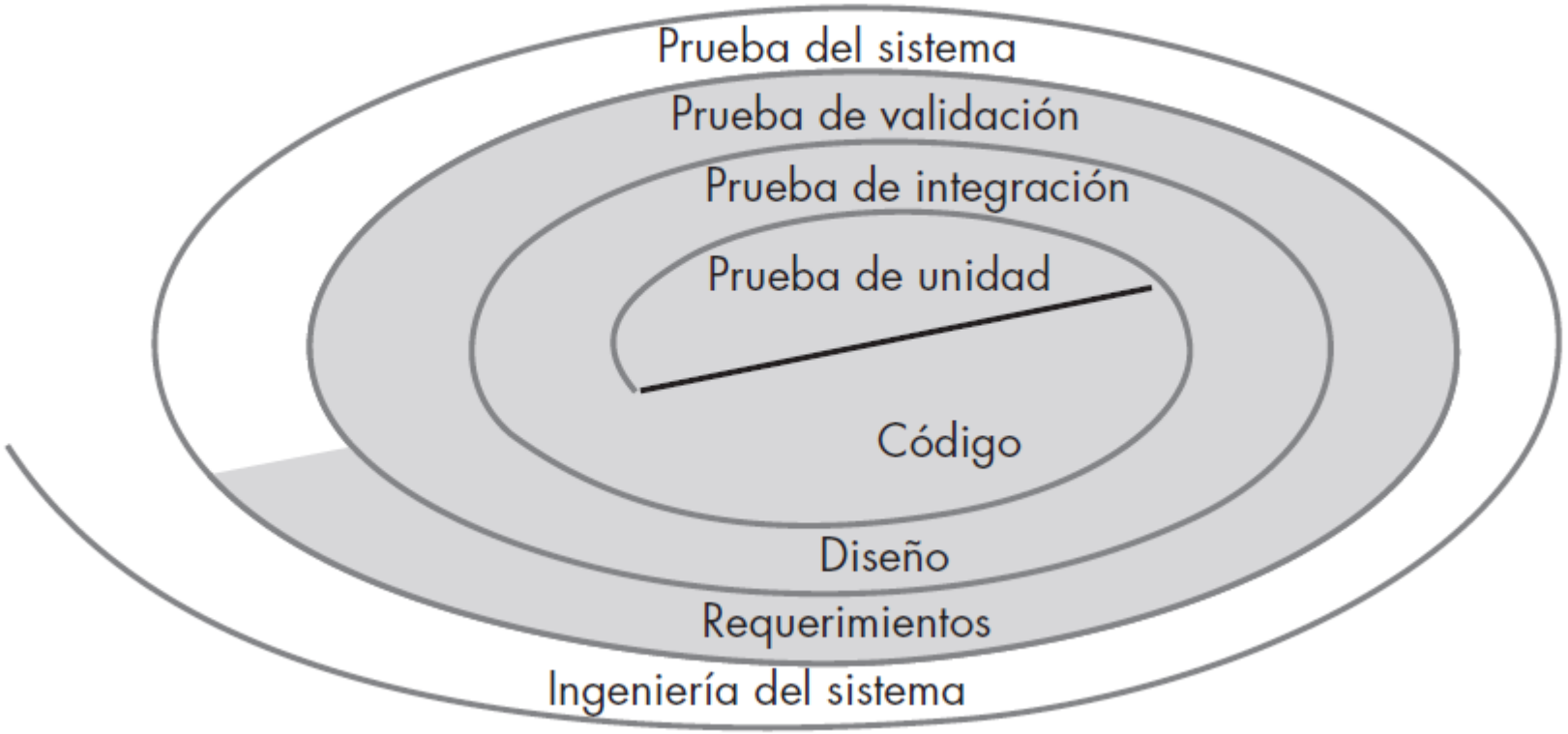
Inspecciones del Software (Técnicas estáticas de verificación):

- Se contrasta estáticamente las diferentes representaciones del sistema (diagramas de requerimientos, diagramas de diseño y código fuente) en búsqueda de defectos.
- No requiere que el código se ejecute
- Debe realizarse durante todo el ciclo de desarrollo.

Las pruebas del Software (Técnicas dinámicas de verificación):

- Se contrasta dinámicamente la respuesta de prototipos ejecutables del sistema con el comportamiento operacional esperado.
- Requiere disponer de prototipo ejecutables y esto sólo es posible en las fases finales del proceso de desarrollo.

Fundamentos de la verificación y validación del software



Estrategia de Pruebas

Fundamentos de la verificación y validación del software

Agenda:

- Verificación
  - La verificación en las diferentes fases del proceso de desarrollo. *Parte I*

Fundamentos de la verificación y validación del software

INTRODUCCIÓN

Fundamentos de la verificación y validación del software

La verificación y validación no puede hacerse en base a intuición, experiencia o suerte.

Se requiere seguir principios y técnicas rigurosas y adecuadas.

Tanto el proceso de desarrollo de software, como los productos generados, deben ser verificados.

Fundamentos de la verificación y validación del software

La actividad debe realizarse en momentos diferentes con objetivos, técnicas y personal diferente.

La producción de software, como cualquier actividad humana, es un proceso sujeto a errores.

Cualquiera que sean los métodos usados, no es posible garantizar que el producto final se comporte siempre como es requerido.

Fundamentos de la verificación y validación del software

La involucración temprana de las pruebas en el ciclo de vida del desarrollo de sistemas ayuda a detectar los defectos con mayor antelación y/o con más facilidad, e incluso ayuda a prevenirlos.

Fundamentos de la verificación y validación del software

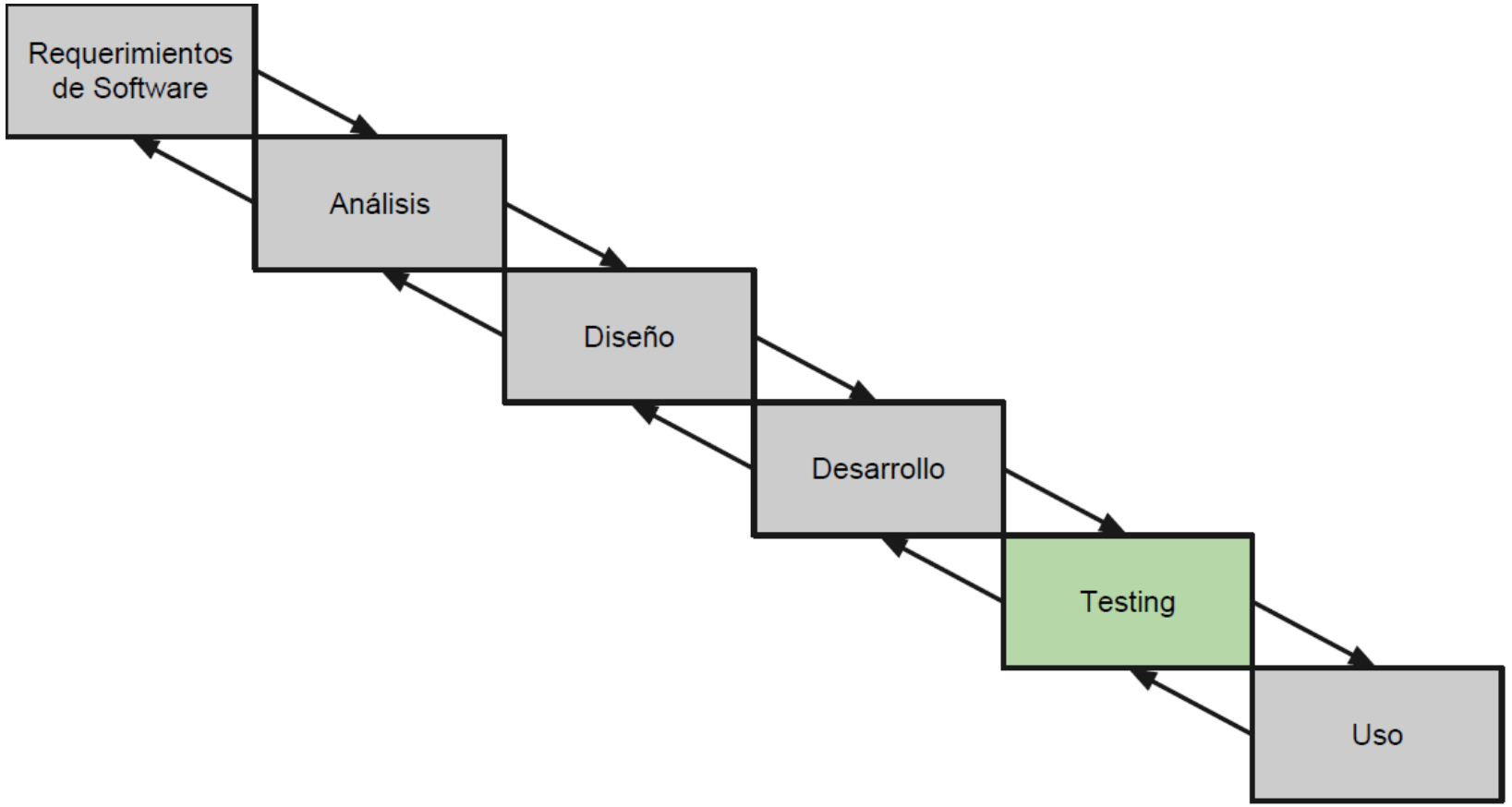
La verificación en las diferentes fases del proceso de desarrollo.

Fundamentos de la verificación y validación del software

El primer modelo de desarrollo de software fue el modelo de Cascada.

Royce, W. W.: "Managing the development of large software systems", IEEE WESCOM, Aug. 1970, pp. 1-9.

Fundamentos de la verificación y validación del software

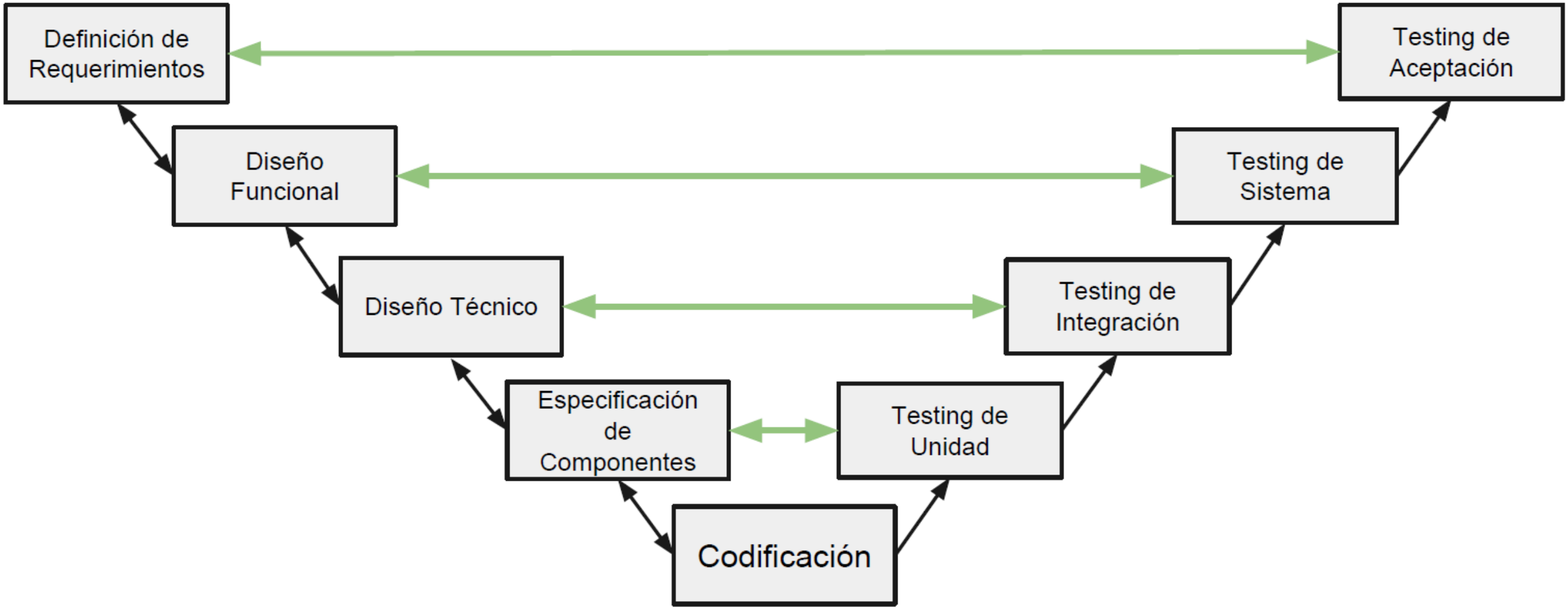


Fundamentos de la verificación y validación del software

Una variante del modelo de cascada es conocida como Modelo V, y da una mejor integración al proceso de testing.

Boehm, B. W.: "Guidelines for Verifying and Validation Software Requirements and Design Specifications", Proceedings of Euro IFIP 1979, pp. 711-719

Fundamentos de la verificación y validación del software



Fundamentos de la verificación y validación del software

En este modelo, el Testing toma el mismo nivel de importancia que el Diseño y el Desarrollo.

En cada nivel de testing (derecha del Modelo V) debemos revisar si se están cumpliendo los requerimientos para dicho nivel de la izquierda del Modelo V.

Fundamentos de la verificación y validación del software

Revisar un producto contra su definición de requerimientos es **Validar** (revisamos si el producto va a servir para lo que se quiere).

Verificar es una acción de bajo nivel, lo hacemos sobre una etapa de desarrollo.

La verificación nos permite determinar si un módulo se implementó de acuerdo a lo especificado por el diseñador.

Fundamentos de la verificación y validación del software

Podemos verificar una clase mirando su código.

Podemos buscar problemas conocidos sin ejecutar el código (por ejemplo, usar = en un IF en Java).

Verificación Estática (no involucra la ejecución del código)

Verificación Dinámica (involucra la ejecución del código) Podemos verificar una clase ejecutándola y comparando la salida de un método con lo esperado.

Fundamentos de la verificación y validación del software

Podemos verificar una clase sin considerar su código.

Podemos verificar que una función F( ) cumple con lo especificado ejecutándola y comparando los resultados.

Fundamentos de la verificación y validación del software

Prueba de Caja Negra (ignoro el código al momento de hacer la verificación)

Prueba de Caja Blanca (genero los casos de test en base a conocimiento que obtengo del código)

Fundamentos de la verificación y validación del software

Ejemplo Implementación

Caso de Prueba. EsPar

Fundamentos de la verificación y validación del software

- Se necesitan "estrategias" para seleccionar casos de prueba "significativos"
- Test Set de Significancia
  - Tiene un alto potencial para descubrir errores
  - La ejecución correcta de estos test aumenta la confianza en el producto
- Más que correr una gran cantidad de casos de prueba nuestra meta en las pruebas debe ser correr un suficiente número de casos de prueba significativos (alto porcentaje de posibilidades de descubrir un error)

Fundamentos de la verificación y validación del software

Prueba de Caja Negra (ignoro el código al momento de hacer la verificación)

Prueba de Caja Blanca (genero los casos de test en base a conocimiento que obtengo del código)

Fundamentos de la verificación y validación del software

Visión de los Objetos a Probar

- **Caja Negra**

Entrada a una caja negra de la que no se conoce el contenido y ver que salida genera

o Casos de prueba - No se precisa disponer del código

o Se parte de los requerimientos y/o especificación

o Porciones enteras de código pueden quedar sin ejercitar

- **Caja Blanca**

A partir del código identificar los casos de prueba interesantes

o Casos de prueba – Se necesita disponer del código

o Tiene en cuenta las características de la implementación

o Puede llevar a soslayar algún requerimiento no implementado

Fundamentos de la verificación y validación del software

Testing de Caja Negra

Ventajas: Encuentra los errores que el usuario ve. No necesita análisis profundo. Independiente de la implementación.

Desventaja: Ineficiente. Intuitivo más que formal.

Fundamentos de la verificación y validación del software

Testing de Caja Blanca

Ventajas: Formal, Eficiente, Metódico.

Desventaja: Puede no ser útil o significativo. Depende de la implementación. Los errores que encuentra pueden no ser importantes.

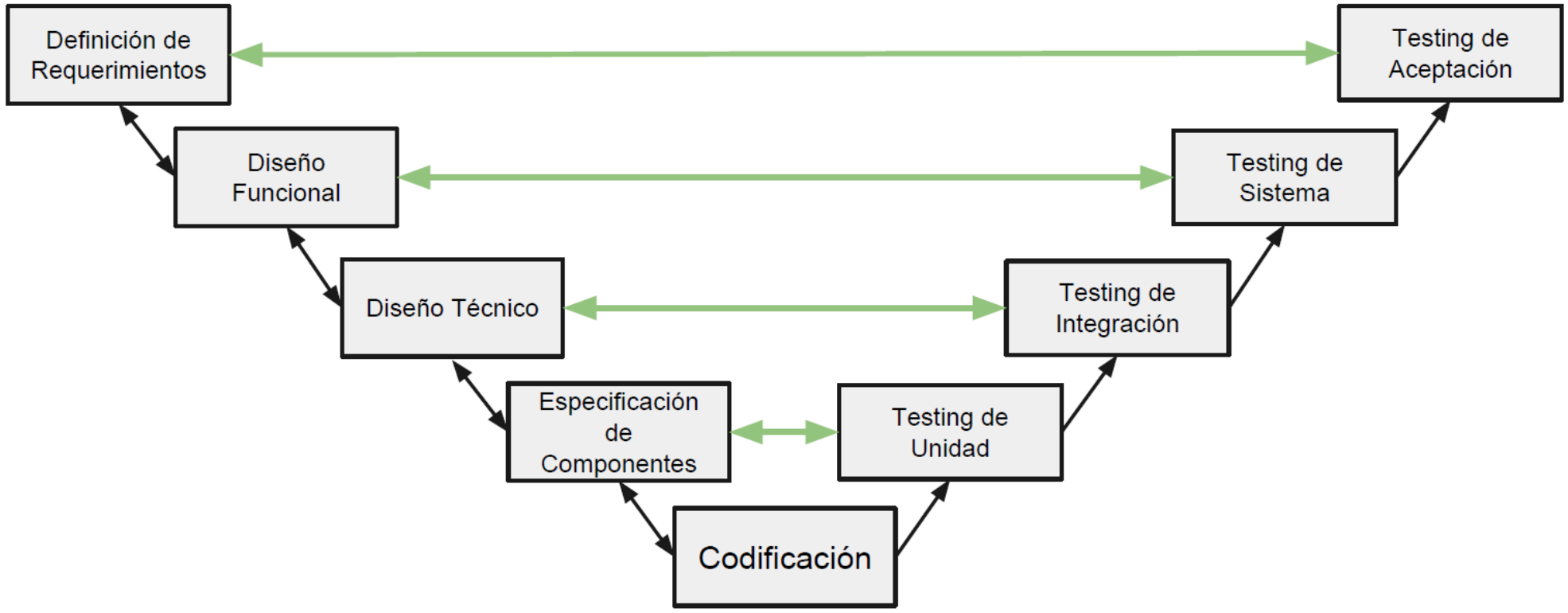
Fundamentos de la verificación y validación del software

Los test de caja blanca detectan entre el 50% y el 75% de los errores, pero son los errores más fáciles de encontrar.

Los test de caja negra detectan un porcentaje menor de errores pero son los errores más problemáticos.

Fundamentos de la verificación y validación del software

Tomando el Modelo V como ciclo de vida, podemos observar diferentes tipos de test de acuerdo al momento del ciclo.



Fundamentos de la verificación y validación del software

Testing de Unidad

Component Testing

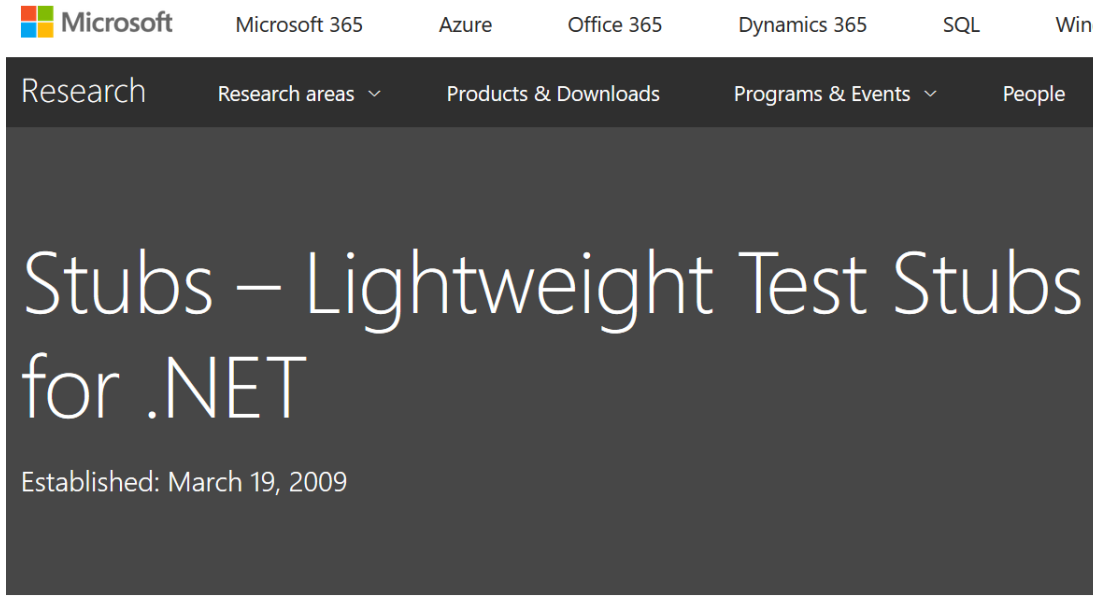
Es el primer test que se hace sobre código. La "Unidad" puede ser una clase, módulo, función, etc, depende del lenguaje/paradigma.

Fundamentos de la verificación y validación del software

- El test de unidad se basa en la especificación del componente. Es fundamental en esta etapa, que cada unidad se testee en **forma aislada**.

- La aislación nos permite estar seguros de que cualquier problema detectada es propio de la unidad. Aislar un componente no es trivial.

Fundamentos de la verificación y validación del software



Stubs is a lightweight framework for .NET that provides test stubs. For interfaces and non-sealed classes, type-safe wrappers are generated that can be easily customized by attaching delegates. Stubs are part of Moles, and work well together with Pex.

Stubs is a lightweight framework for **test stubs in .NET** that is entirely based on delegates. Stubs may be used on interfaces, abstract classes or non-sealed classes. Stubs was designed provide a minimal overhead to the Pex white box analysis, and Stubs supports the Code Contracts runtime rewriter and encourage the programmatic models rather than record/replay tests.

Aislar el código probado con Microsoft Fakes

04/11/2016 • 13 minutos de lectura • Colaboradores

Microsoft Fakes ayuda a aislar el código que se está probando mediante la sustitución de otros elementos de la aplicación con *código auxiliar* o *correcciones de compatibilidad (shim)*. Se trata de pequeños fragmentos de código que están bajo el control de las pruebas. Al aislar código para pruebas, sabe que, en caso de error, la causa está localizada ahí y no en alguna otra parte. El código auxiliar y las correcciones de compatibilidad (shims) también permiten probar el código aunque no funcionen otras partes de la aplicación todavía.

Fakes tiene dos versiones:

- El código auxiliar reemplaza a una clase por un pequeño sustituto que implementa la misma interfaz. Para utilizar código auxiliar, tiene que diseñar la aplicación para que cada componente dependa únicamente de interfaces y no de otros componentes. (Por "componente" se entiende una clase o grupo de clases diseñadas y actualizadas a la vez y que suelen estar contenidas en un ensamblado).

- Una corrección de compatibilidad (shim) modifica el código compilado de la aplicación en tiempo de ejecución para que, en lugar de realizar una llamada de método especificada, ejecute el código shim que proporciona la prueba. Las correcciones de compatibilidad (shims) se pueden usar para reemplazar las llamadas a ensamblados que no se pueden modificar, como los ensamblados .NET.

Fundamentos de la verificación y validación del software

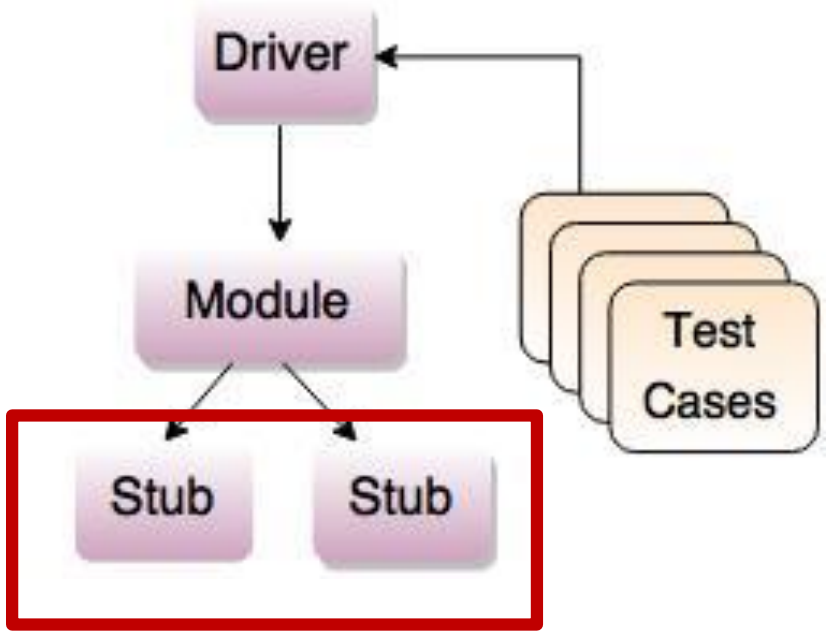
- Este nivel de pruebas lo realiza un tester trabajando en conjunto con el desarrollador, o lo hace el desarrollador mismo.

- A veces, para testear una componente, no alcanza con una única línea de código. En ocasiones, hay que hacer casi otra unidad.

- A esto último se le llama **Test Driver. (Tester que maneja al objeto de la prueba)**

Fundamentos de la verificación y validación del software

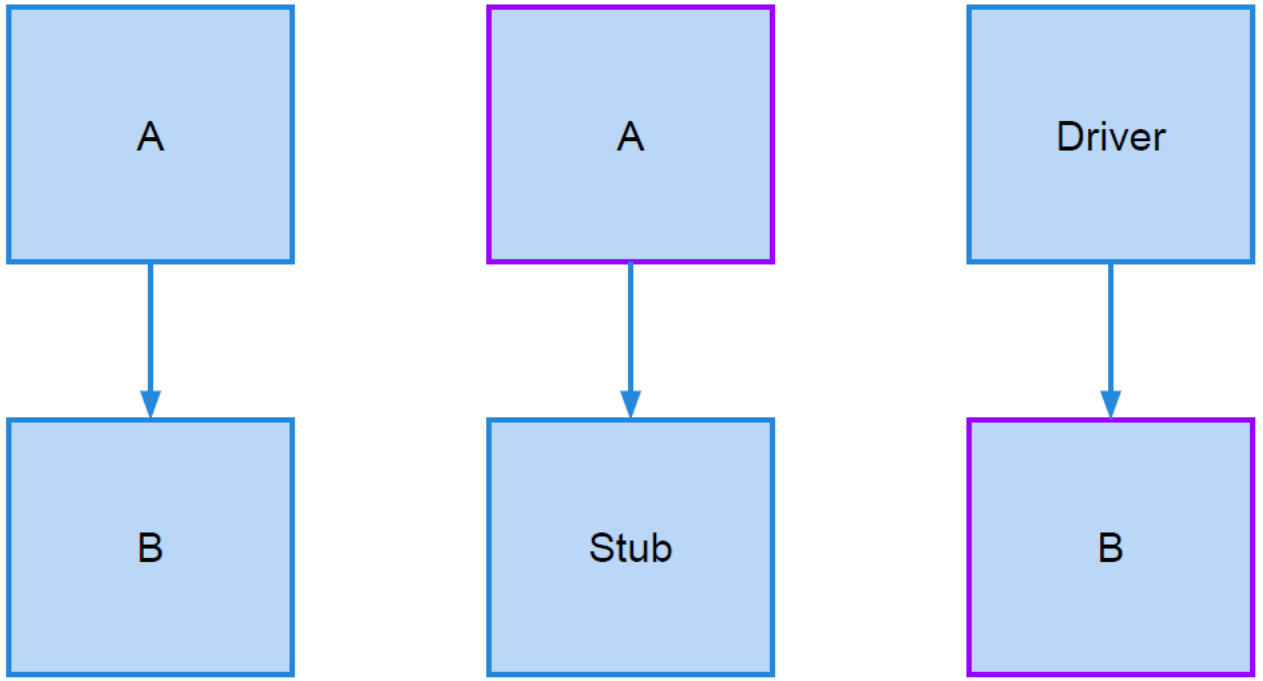
Cuando el componente bajo prueba debe llamar a otro componente, para mantener la aislación, se reemplaza la segunda por una componente tonta o **Stub**.



Fundamentos de la verificación y validación del software

Pruebas de Unidad

Stub
Driver



Fundamentos de la verificación y validación del software

El objetivo de este tipo de pruebas es asegurar funcionalidad y completitud. "El componente hace todo lo que requieren, ni más ni menos, y lo hace bien".

Fundamentos de la verificación y validación del software

También se busca **solidez** y **robustez**.

El componente es sólido, "el componente no se rompe".

El componente es robusto, "si el componente se rompe no hace un desastre".

Fundamentos de la verificación y validación del software

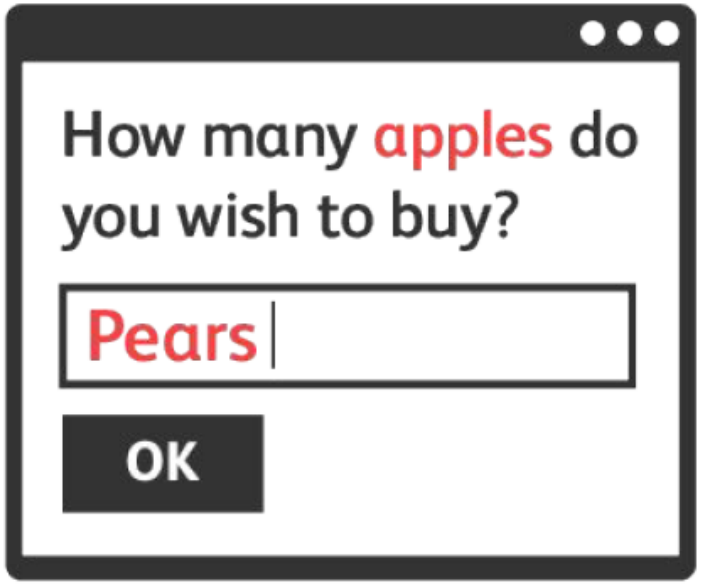
Prueba de unidad

"El componente hace lo que tiene que hacer" - Validación

"El componente lo hace bien" - Verificación

Fundamentos de la verificación y validación del software

Cuando se testea solidez y robustez aparecen los test negativos. Estos son test que buscan romper cosas.



Fundamentos de la verificación y validación del software

- El test de unidad suele ser un test de caja blanca. El desarrollador conoce el código y puede hacer test en base al mismo.
- Se puede hacer también test de caja negro, teniendo acceso al código.

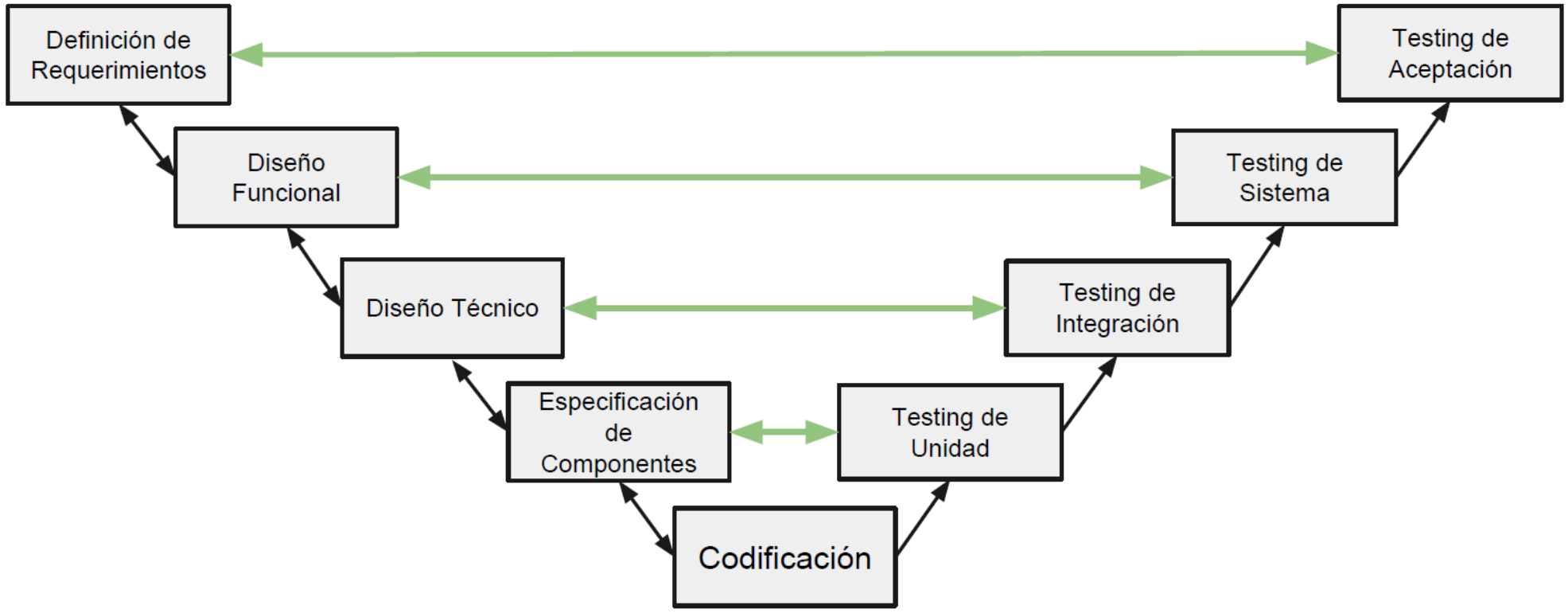
Fundamentos de la verificación y validación del software

- Los test de unidad se puede automatizar.

- A esto se le conoce como “test-first programming” o “test-driven development”

Fundamentos de la verificación y validación del software

Tomando el Modelo V como ciclo de vida, podemos observar diferentes tipos de test de acuerdo al momento del ciclo.



Fundamentos de la verificación y validación del software

TERMINOLOGÍA

Fundamentos de la verificación y validación del software

Se define como **objeto de prueba**, al elemento que estamos probando.

La ejecución de un objeto de prueba se realiza con **datos de prueba.**

La **administración de las pruebas** incluye el planeamiento,

implementación, documentación y análisis de la prueba.

El proceso de pruebas incluye la **administración** de las mismas.

Fundamentos de la verificación y validación del software

Se define como **suite de prueba** a la ejecución de uno o más casos de prueba.

Un **caso de prueba** contiene el objeto de la prueba, condiciones de ejecución, parámetros de entrada y salida esperada.

La **concatenación de casos de prueba**, de modo que la entrada de una prueba sea la salida de la anterior forman un escenario de prueba.

Gracias por su atención