Ingeniería del Software I

2º

Tema 6
Pruebas de software

Universidad Francisco de Vitoria
UFV Madrid

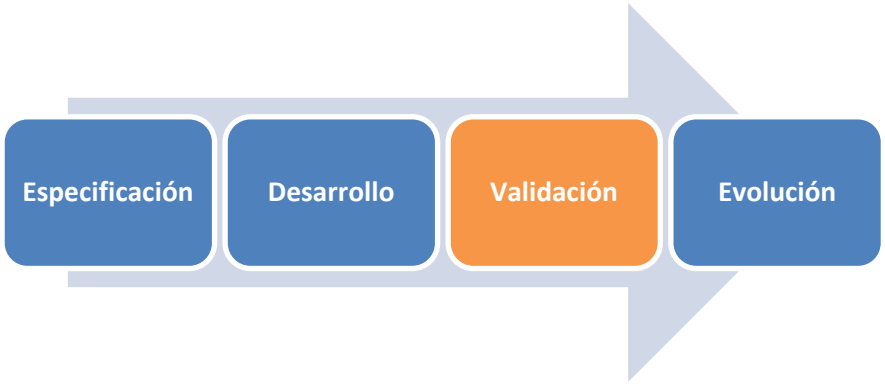
Grado en Ingeniería Informática
Escuela Politécnica Superior

Temas

Universidad Francisco de Vitoria UFV Madrid

Objetivos

- Pruebas de desarrollo
- Desarrollo basado en pruebas
- Pruebas de lanzamiento
- Pruebas de usuario



Prueba de software

Universidad Francisco de Vitoria UFV Madrid

- La prueba intenta demostrar que un programa **hace lo que se pretende hacer** y **descubre los defectos** del programa antes de su puesta en marcha.
- Al probar el software, se ejecuta un programa que utiliza los datos inventados/artificiales.
- Hay que **comprobar los resultados de la prueba** que se ejecuta para buscar errores, anomalías o información acerca de los atributos no funcionales del programa.
- *Las pruebas pueden revelar sólo la presencia de errores, mas no su ausencia. (Dijkstra)*
  - No va a probar que funcione siempre.
- **El valor de un equipo de desarrollo está en su capacidad de detectar ERRORES**

Metas de la prueba de software

Universidad Francisco de Vitoria UFV Madrid

- Demostrar al desarrollador y al cliente que el software cumpla con sus requisitos (Cumple requisitos. Lleva a la **prueba de validación**)
  - **Para el software personalizado**, esto significa que debe haber *por lo menos una prueba por cada requisito* en el documento de requisitos. Para los productos de **software genérico**, significa que debe haber *pruebas para todas las funciones del sistema*, además de combinaciones de estas características, que se incorporarán en la versión del producto.
- Descubrir las situaciones en las que el comportamiento del software es incorrecta, indeseable o no, se ajuste a su especificación. (No hay errores. Lleva a la **prueba de defectos**)
  - **Prueba de defectos se ocupa de erradicar el comportamiento del sistema no deseado**, tales como fallos del sistema, las interacciones no deseadas con otros sistemas, cálculos incorrectos y corrupción de datos.

Validación y pruebas de defectos

Universidad Francisco de Vitoria UFV Madrid

El **primer objetivo** conduce a las **pruebas de validación**.

- Se espera que el sistema **realice correctamente el uso** de un determinado conjunto de casos de prueba que reflejen el uso esperado del sistema.

El **segundo objetivo** conduce a **pruebas de defectos**.

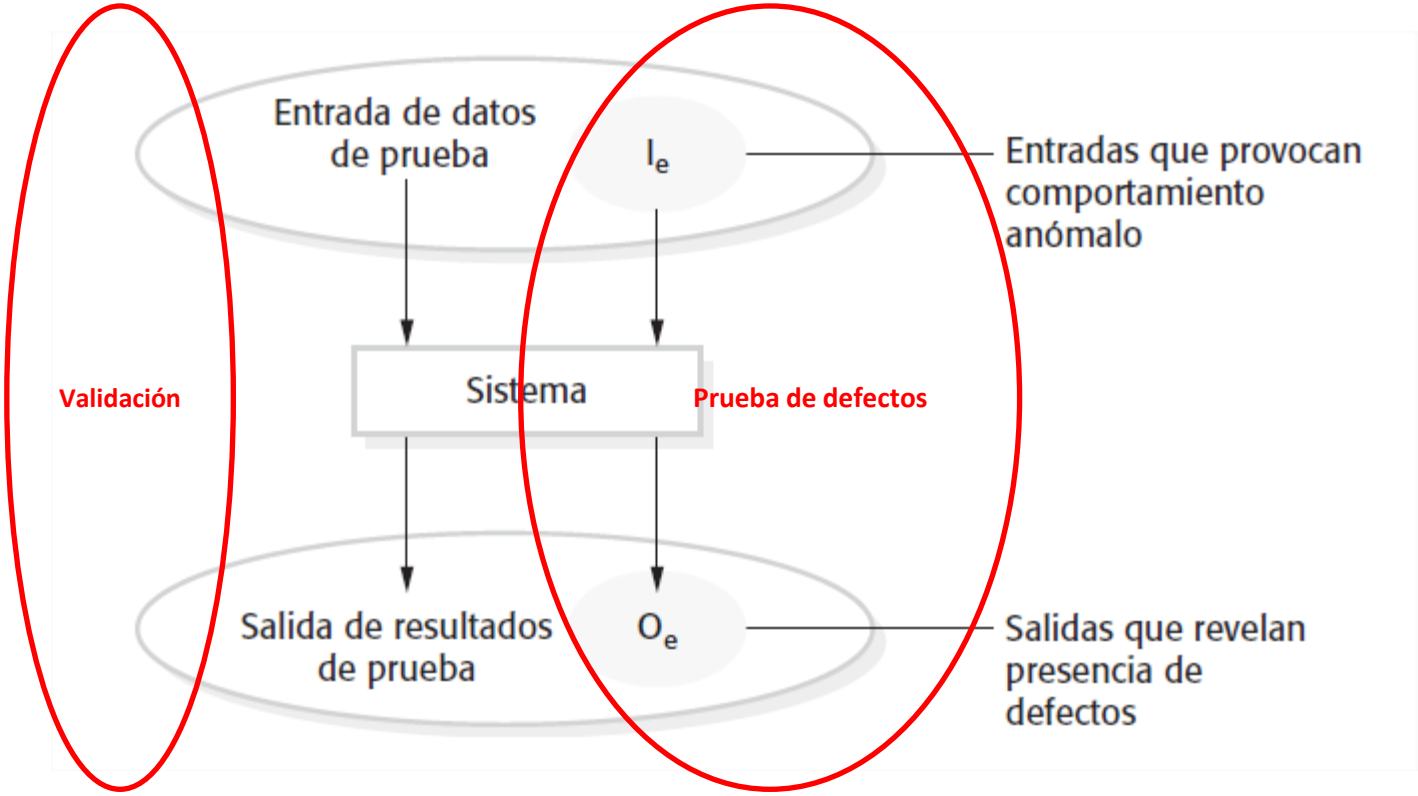
- Los casos de prueba están **diseñados para exponer los defectos**. Los casos de prueba en pruebas de defectos pueden ser deliberadamente confusos y no tienen por qué reflejar cómo se utiliza normalmente el sistema.

Metas del proceso de prueba

Universidad Francisco de Vitoria UFV Madrid

- **Las pruebas de validación**
  - Para demostrar al desarrollador y al cliente del sistema que el software cumpla con sus requisitos
  - Una prueba exitosa muestra que el **sistema esté funcionando** como tiene que ser.
- **Pruebas de defectos**
  - Para descubrir los fallos o defectos en el software donde su comportamiento es incorrecto o que no esté conforme con su especificación
  - Una prueba con éxito es una prueba que hace que **el sistema se realice correctamente** y así expone un defecto en el sistema.

Un modelo entrada-salida de prueba del programa



Verificación vs validación

Universidad Francisco de Vitoria UFV Madrid

- Los procesos de comprobación comienzan tan pronto como están disponibles los requisitos y continúan a través de todas las etapas del proceso de desarrollo.

- **Validación**: "Estamos construyendo el producto correcto".
  ¿Cumple las expectativas del cliente?
  - Proceso más general.
  - Finalidad: El software debe hacer lo que el usuario realmente necesita, que cumpla las **expectativas** del cliente.
- **Verificación**: "Estamos construyendo bien el producto, estamos construyendo el producto correctamente".
  - **¿Cumple los requerimientos?**
  - Finalidad: El software debe ejecutarse conforme a su especificación, que cumpla con su **funcionalidad** y con los requisitos no funcionales establecidos.

Confianza V y V (validación y verificación)

Universidad Francisco de Vitoria UFV Madrid

- **Objetivo de V & V** es establecer la confianza de que el sistema es "adecuado".
- El **nivel de confianza** adquirido depende de tanto del propósito del sistema, y las expectativas de los usuarios del sistema, como del entorno del mercado para el sistema:
    - **Propósito** del Software. Cuanto más crítico sea el software, más **confiable,** que funcione todo correctamente.
    - **Expectativas** de los usuarios. Debido a su experiencia de software no confiable y errores previos, muchos usuarios tienen pocas expectativas de calidad.
    - Entorno del **mercado**. Cuando un sistema se comercializa, los vendedores han de tener en cuenta los productos competidores, el precio de la demanda y la fecha de entrega estimada.

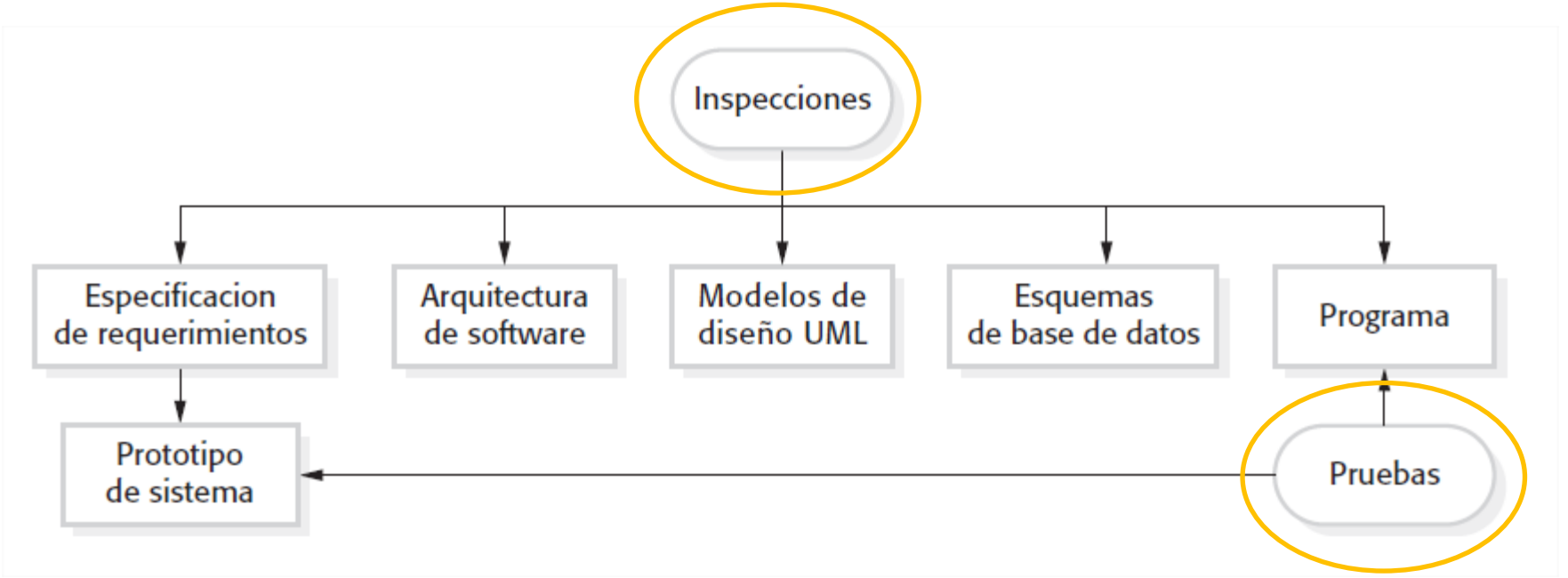
Inspecciones y pruebas

Universidad Francisco de Vitoria UFV Madrid

- **Las inspecciones de software (pruebas estáticas)**: no es necesario ejecutar el software para verificarlo.
    - Analizan y comprueban los **requerimientos del sistema**, modelos de diseño, código fuente y pruebas propuestas.
- **Pruebas de software**: observan el comportamiento del producto **(verificación dinámica)**
    - El sistema se ejecuta con **datos de prueba y se observa su comportamiento** operativo.

Inspecciones y pruebas

Universidad Francisco de Vitoria UFV Madrid



Ventajas de las inspecciones

Universidad Francisco de Vitoria UFV Madrid

- **Errores ocultos**. Durante las pruebas, los errores pueden enmascarar (ocultar) otros errores. Debido a que la inspección es un proceso estático, no hay que preocuparse por las interacciones entre los errores. Una sola sesión de inspección descubrirá muchos errores en un sistema.
- **Versiones incompletas**. Las versiones incompletas de un sistema pueden ser inspeccionadas sin costes adicionales. Si un programa está incompleto, es necesario crear equipos de prueba especializados para poner a prueba las partes disponibles. Esto genera costes.
- **Cumplimiento de estándares**. Además de la búsqueda de defectos del programa, la inspección también puede considerar atributos de calidad más amplios de un programa como: el cumplimiento de estándares, la portabilidad y mantenibilidad.
  - Pueden buscarse ineficiencias, algoritmos inadecuados, estilos de programación imitados que hagan el sistema difícil de mantener.

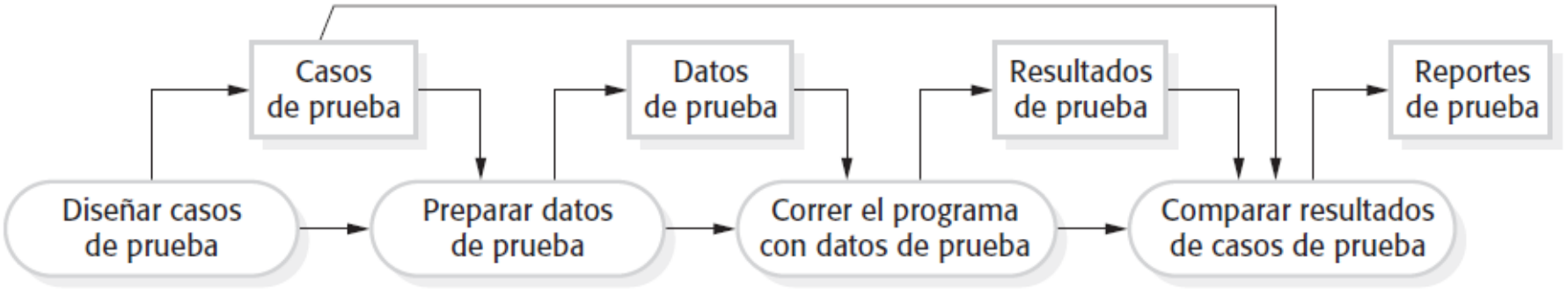
Inspecciones y pruebas

Universidad Francisco de Vitoria UFV Madrid

- *Fagan: "más del 60% de los errores de un programa se detectan mediante inspecciones informales del programa"*.
- Las inspecciones y las pruebas son complementarias y no se oponen a las técnicas de verificación.
- Ambas deben ser utilizados durante el proceso de V & V.
- Las inspecciones pueden comprobar la conformidad con una especificación, pero no la conformidad con los requisitos reales del cliente.
- Las inspecciones no pueden comprobar las características no funcionales tales como rendimiento, usabilidad, etc.

Un modelo del proceso de prueba de software

Universidad Francisco de Vitoria UFV Madrid



Etapas de la prueba

Universidad Francisco de Vitoria UFV Madrid

- Por lo general, un sistema de software comercial debe pasar por **tres etapas de pruebas**:
  - Pruebas de desarrollo, donde el sistema se prueba durante el desarrollo para descubrir los errores (bugs) y defectos.
  - Versiones de prueba - **Alfa**, en donde un equipo de pruebas independiente, testea una versión completa del sistema antes de presentarlo a los usuarios finales.
  - Pruebas de usuario - **Beta**, donde los usuarios reales o usuarios potenciales de un sistema, prueban el sistema en su propio entorno. Ejemplo: Dpto Marketing.

1 - Pruebas de desarrollo

Universidad Francisco de Vitoria UFV Madrid

- **Pruebas de desarrollo** incluye todas las actividades de pruebas que se llevan a cabo por el equipo de desarrollo del sistema.
  - **Pruebas unitarias**, donde se ponen a prueba las unidades de programa individuales o clases de objetos. Prueba de la unidad debe centrarse en probar la funcionalidad de los objetos o métodos.
  - **Pruebas de componentes o de integración**, donde se integran varias unidades individuales para crear componentes compuestos. Deberían centrarse en las interfaces de componentes de prueba.
  - **Pruebas del sistema**, donde algunos o todos los componentes de un sistema están integrados y el sistema se pone a prueba en su conjunto. Las pruebas del sistema debería centrarse en las interacciones de los componentes de prueba.

1.1 Prueba de unidad

Universidad Francisco de Vitoria UFV Madrid

- Prueba de la unidad es el proceso de probar los componentes individuales de forma aislada.
- Se trata de un proceso de pruebas de defectos aplicado al desarrollo.
- Las unidades pueden ser:
  - Las **funciones individuales o métodos** dentro de un objeto.
  - Las **clases de objetos** con varios atributos y métodos
  - **Componentes** compuestos con interfaces definidas usados para acceder a su funcionalidad.

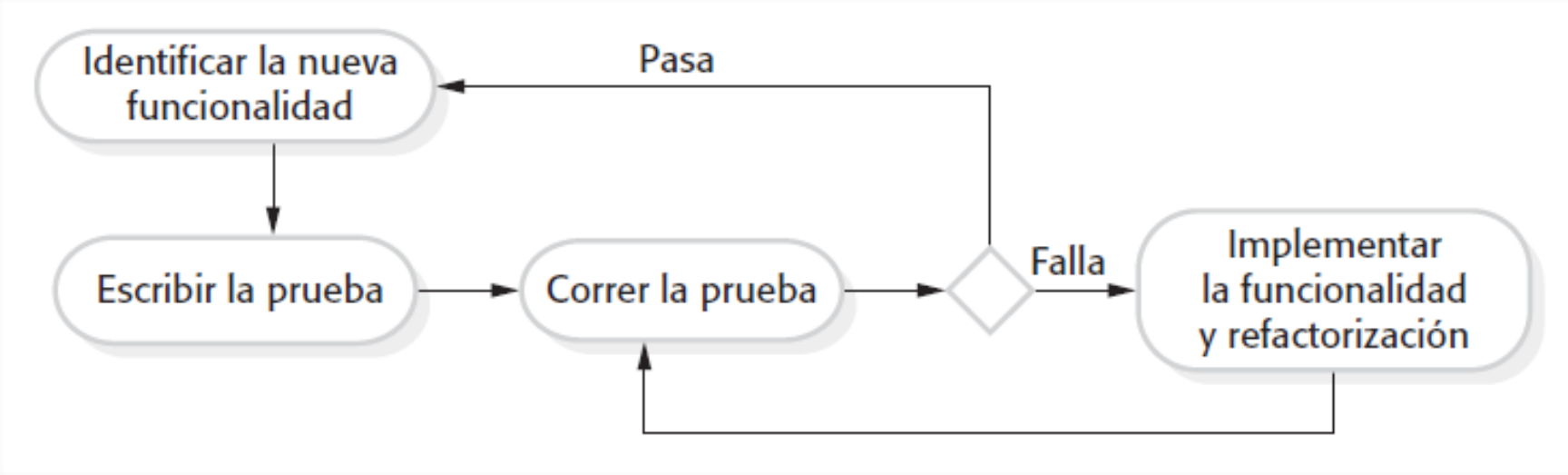
Desarrollo dirigido por pruebas

Universidad Francisco de Vitoria UFV Madrid

- **Desarrollo incremental + Prueba incremental**
- **Desarrollo dirigido por pruebas (TDD)** es un enfoque en el que se entrelazan el **desarrollo de pruebas y el desarrollo de código**.
- **Desarrolla código de forma incremental**, junto con una prueba para ese incremento.
  - No se pasa al siguiente incremento hasta que el código que se desarrolla pasa su prueba.
- **TDD** fue presentado como parte de los métodos ágiles como **Extreme Programming**. Sin embargo, también se puede utilizar en los procesos de desarrollo basados en un plan.

Desarrollo dirigido por pruebas

Universidad Francisco de Vitoria UFV Madrid



Actividades del proceso de TDD

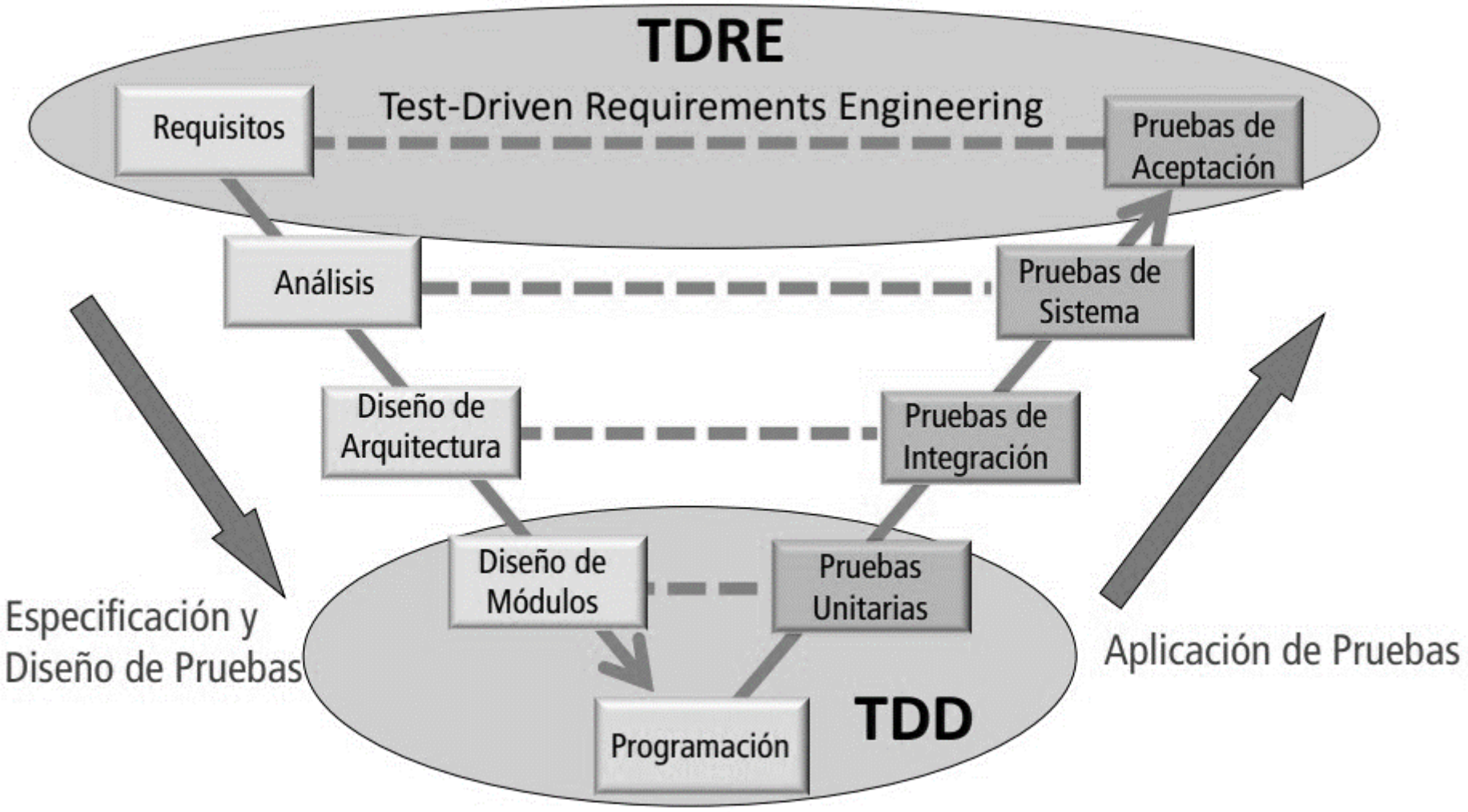
Universidad Francisco de Vitoria UFV Madrid

1. Se comienza por identificar el **incremento** de la funcionalidad que se requiere. Normalmente, debería ser pequeño y aplicable en unas pocas líneas de código.
2. Se **escribe una prueba** para esta funcionalidad y aplica esto como una prueba automatizada.
3. Se **ejecuta la prueba**, junto con todas las demás pruebas que se han implementado. Inicialmente, no se ha implementado la funcionalidad de modo que la nueva prueba fallará.
4. Se **implementa la funcionalidad** y vuelve a **ejecutar la prueba**.
5. Una vez que todas las pruebas se ejecutan correctamente, se pasa a la aplicación de la siguiente parte de la funcionalidad.

- Por tanto, las **pruebas se introducen en un programa independiente** que ejecuta las pruebas y **llama al sistema que se prueba**.
- El **desarrollo dirigido por pruebas** ayuda a los programadores a aclarar sus ideas acerca de lo que realmente debe hacer una parte de código.

Actividades del proceso de TDD

Universidad Francisco de Vitoria UFV Madrid



Beneficios del desarrollo basado en pruebas

Universidad Francisco de Vitoria UFV Madrid

- Uno de los beneficios más importantes del desarrollo dirigido por pruebas es que **reduce los costes de las pruebas de regresión**.

- El desarrollo dirigido por pruebas resulta ser un enfoque de éxito para proyectos de pequeña o mediana dimensión.

¿Pruebas de regresión=pruebas automatizadas=fuera las pruebas manuales?

- El concepto de pruebas de regresión como tal, no implica que tengan que ser pruebas automatizadas, es algo que se **tiende a realizar**, para ejecutarlas frecuentemente.
- Tener pruebas de regresión no es una garantía de que todo funciona perfectamente.
- Son pruebas que **ya hemos ejecutado antes, y que ejecutamos periódicamente después de algún cambio**, para asegurarnos de que errores que ya detectamos antes y resolvemos no vuelven a aparecer. (Javier Garzás)

https://www.javiergarzas.com/2014/06/pruebas-de-regresion.html

Beneficios del desarrollo basado en pruebas

Universidad Francisco de Vitoria UFV Madrid

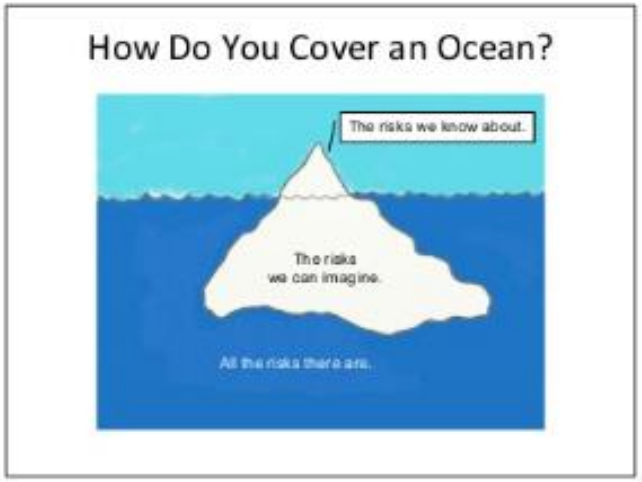
Michael Bolton

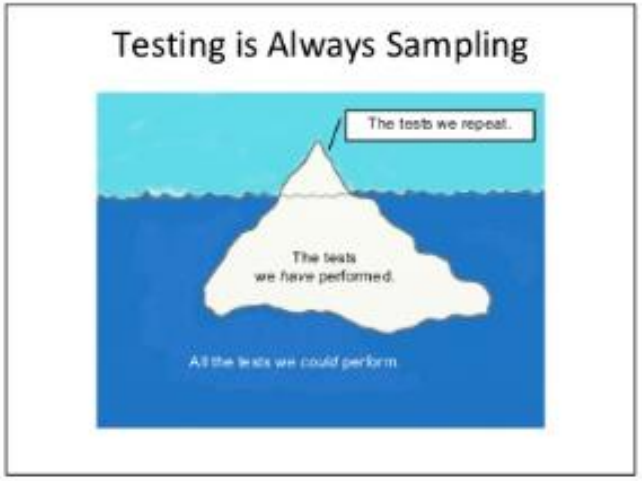
http://www.developsense.com

Las pruebas de regresión, forman parte de esta **punta del iceberg**.

Hay que realizar las pruebas de regresión **de forma complementaria** a las pruebas manuales, pensadas o planificadas con el fin de abarcar la totalidad del sistema y descubrir más errores (en el resto del iceberg).

Es decir, si se vuelve a ejecutar un caso, que antes fallaba, en una nueva versión en la que se ha corregido el error, **esta versión debería pasar la prueba** con lo que nos aseguramos no tener que volver a modificar una versión anterior para resolver el mismo fallo

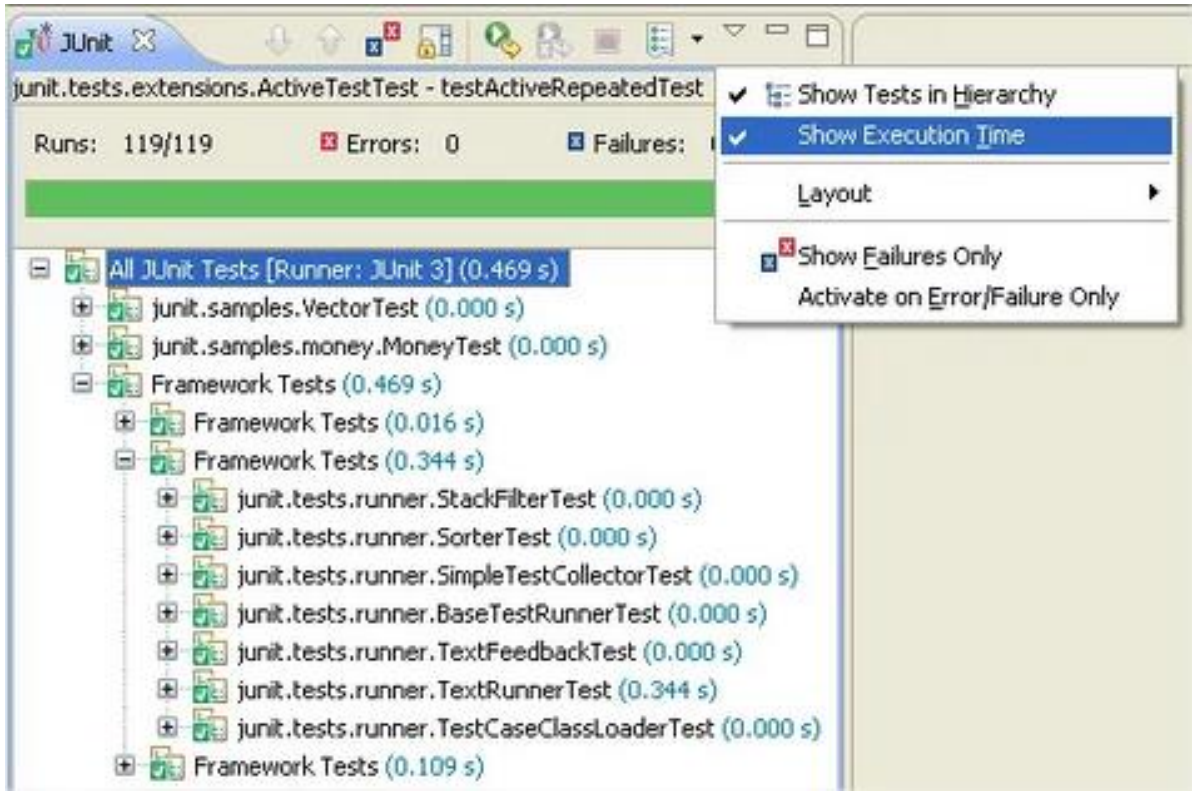




Beneficios del desarrollo basado en pruebas

Universidad Francisco de Vitoria UFV Madrid

TDD fue creado por Kent Beck (quien también inventó Extreme Programming y JUnit)



Pruebas de versión. Release Testing

Universidad Francisco de Vitoria UFV Madrid

- Es el proceso de **poner a prueba una versión particular** de un sistema que se pretende usar fuera del equipo de desarrollo.
- Existen dos **distinciones entre las pruebas de versión y las pruebas de sistema** durante el proceso de desarrollo:
  - Un **equipo independiente que no** intervino en el desarrollo del sistema debe ser el responsable de las pruebas de versión.
  - Las **pruebas del sistema por parte del equipo de desarrollo** deben enfocarse en el descubrimiento de bugs en el sistema (**pruebas de defecto**)
- Las pruebas de versión, son un proceso de caja negra, donde las pruebas se derivan a partir de la especificación del sistema.
- El sistema se trata como una caja negra cuyo comportamiento sólo puede determinarse por el estudio de entradas y salidas relacionadas.

Pruebas de escenario. Scenery tests

Universidad Francisco de Vitoria UFV Madrid

- Un escenario es una historia en la que se describe una forma de utilización del sistema. Un contexto.
- Las pruebas de escenario consisten en crear una combinación de pruebas equivalentes a un escenario real.
- Los escenarios deben ser realistas y los usuarios reales del sistema tienen que relacionarse con ellos.
- Una prueba de escenario requiere una **narrativa que sea creíble y bastante compleja**. Ha de ser **fácil de evaluar**.
- Ejemplo de características que se pueden poner a prueba:
  - Autenticación por sesión en el sistema.
  - Recuperación de registros y modificación.
- Cuando se usa un enfoque basado en escenarios, se ponen a prueba por lo general, varios requerimientos dentro del mismo escenario.
  - Por tanto, además de comprobar requerimientos individuales, también demuestra las combinaciones de requerimientos no causan problemas.
- **Problema**: es difícil diseñar pruebas de escenario que cubran todos los requisitos

Pruebas de rendimiento. Pruebas de carga

Universidad Francisco de Vitoria UFV Madrid

- Una vez integrado el sistema, es posible incluir pruebas de las propiedades emergentes de un sistema, como el rendimiento y la confiabilidad.
- Un **perfil operativo** es un conjunto de pruebas que reflejan la mezcla real de trabajo que manejará el sistema.
- **Pruebas de carga, o de esfuerzo**: estresar el sistema al hacer demandas que estén fuera de los límites de diseño del software. Tienen dos funciones:
  - Prueba el comportamiento de fallo del sistema.
  - Fuerza al sistema y puede hacer que salgan a la luz defectos que no se descubrían normalmente.
- Las pruebas de esfuerzo son relevantes para los sistemas distribuidos basados en redes de procesadores. Muestran con frecuencia degradación cuando se cargan en exceso.

Pruebas de usuario

Universidad Francisco de Vitoria UFV Madrid

- Las pruebas de usuario o cliente es una etapa en el proceso de pruebas en el que los **usuarios o los clientes proporcionan información y asesoramiento** sobre las pruebas del sistema.
- La prueba de usuario es esencial, incluso cuando el sistema completo y pruebas de liberación se han llevado a cabo.
    - La razón de esto es que las influencias del entorno de trabajo del usuario tienen un efecto importante en la fiabilidad, rendimiento, facilidad de uso y robustez de un sistema. Estos no pueden ser replicados en un entorno de prueba.

Tipos de prueba de usuario

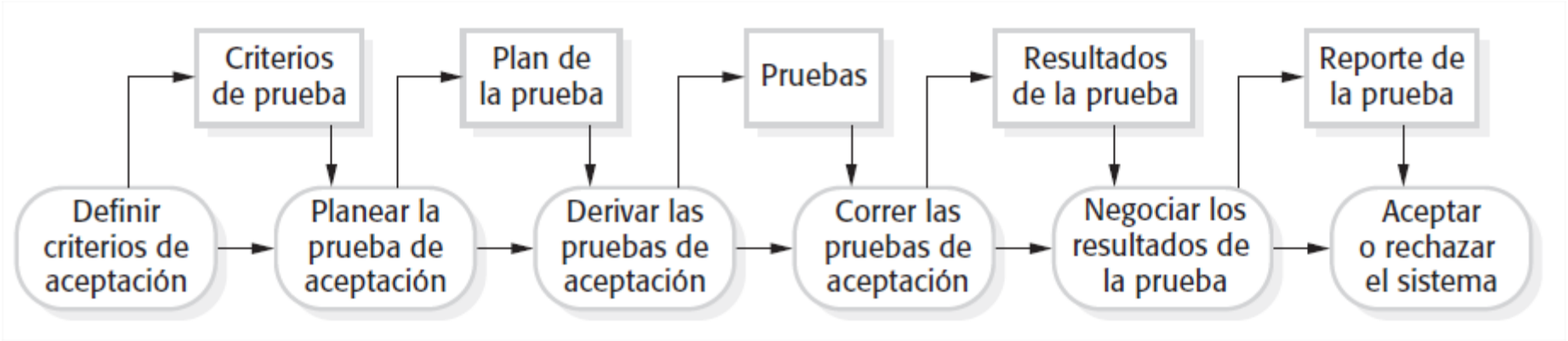
Universidad Francisco de Vitoria UFV Madrid

Lo ideal sería que el **usuario acompañara en cada parte del proceso de prueba**, sobre todo cuando es difícil tener los datos reales del sistema:

- Pruebas alfa
  - Es la primera versión del programa, la cual es enviada a los verificadores para probarla. Algunos equipos de desarrollo utilizan el término alfa informalmente para referirse a una fase donde un producto todavía es inestable pero contempla la mayoría de los requisitos.
  - Los usuarios del software trabajan con el equipo de desarrollo para poner a prueba. Principalmente utilizadas para sistemas empaquetados.
- Pruebas beta
  - Se llevan a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente generalmente. Así, la prueba beta es una aplicación real del sistema en un entorno que no puede ser controlado por el desarrollador.
- Pruebas de aceptación
  - Los clientes prueban un sistema para decidir si esta o no está listo para ser aceptado por los desarrolladores de sistemas y desplegado en el entorno del cliente.

El proceso de las pruebas de aceptación

Universidad Francisco de Vitoria UFV Madrid



Etapas en el proceso de las pruebas de aceptación

Universidad Francisco de Vitoria UFV Madrid

Existen seis etapas en el proceso de pruebas de aceptación:

1. Definir los criterios de aceptación
2. Planificar las pruebas de aceptación
3. Derivar las pruebas de aceptación
4. Ejecutar las pruebas de aceptación
5. Negociar resultados de las pruebas
6. Rechazar/aceptar el sistema

Los métodos ágiles y pruebas de aceptación

Universidad Francisco de Vitoria UFV Madrid

- Una prueba de aceptación tiene como propósito demostrar al cliente el cumplimiento de un requisito del software.
- En los métodos ágiles, el **usuario/cliente es parte del equipo de desarrollo** y es responsable de la toma de decisiones sobre la aceptabilidad del sistema.
- Las **pruebas son definidas por el usuario/cliente y se integran con otras pruebas las que se ejecutan automáticamente** cuando se realizan cambios.
- No hay aceptación independiente del proceso de prueba.
- El principal problema aquí es si esta o no el usuario incorporado es "típico" y puede representar los intereses de todos los actores del sistema.

https://www.youtube.com/watch?v=Hzx3tf_s9YU

Testing Ágil (Keynote de apertura de Testing Uruguay 2016)