Testing Software

Pruebas y mantenimiento de software

Lunes de 8:00 a 10:00

en CReCE

Prof. José Antonio Cervantes Álvarez

antonio.alvarez@academicos.udg.mx

Unidad I. Fundamentos de pruebas de software

1.5 El proceso fundamental de las pruebas.

- 1.5.1 Planeación y control de pruebas.
- 1.5.2 Análisis y diseño de pruebas
- 1.5.3 Implementación y ejecución de pruebas.
- 1.5.4 Evaluación de pruebas e informe.
- 1.5.5 Cierre de las actividades de prueba.

1.6 La psicología de las pruebas.

1.7 Principios generales de las pruebas.

1.8 La ética del ingeniero de pruebas.

Repaso

- ¿Qué son las pruebas de software?
- ¿Cuál es el objetivo de las pruebas de software?
- ¿Qué es un fallo en el software y cuándo ocurre?
- ¿Qué es un defecto en el software y cuándo ocurre?
- ¿Qué es un error en el software y cuándo ocurre?

Repaso

- ¿Cuáles son las características o atributos relacionados con la calidad del software?

El proceso fundamental de las pruebas

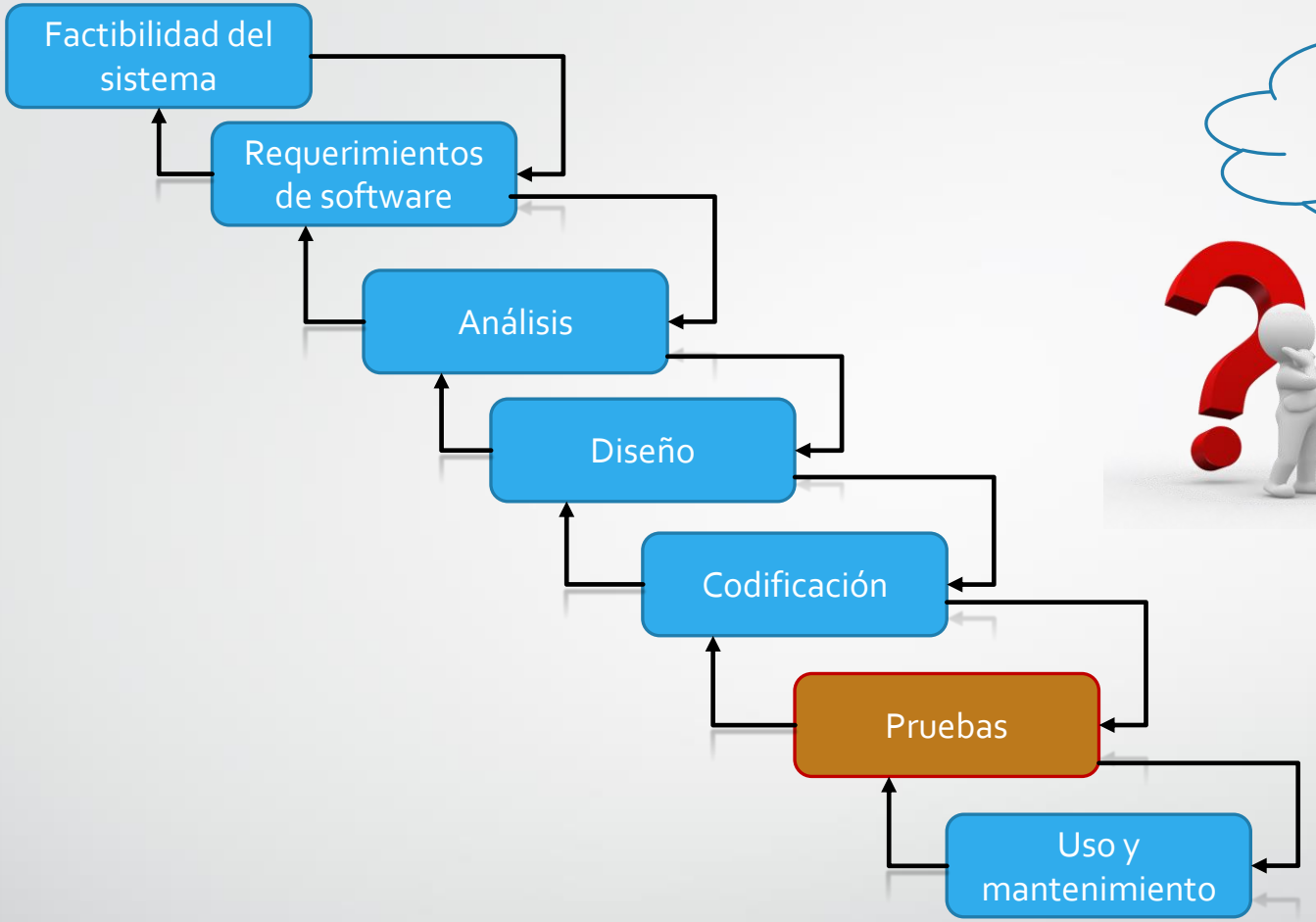
- **Breve repaso de los modelos usados en el ciclo de vida del software.**
  - Para lograr un esfuerzo de desarrollo de software estructurado y controlado es necesario utilizar modelos de desarrollo de software. Algunos de los modelos más utilizados son:
    - Modelo en cascada.
    - Modelo general V.
    - Modelo Alemán V XT.
    - Modelo en espiral.
    - Modelo incremental o evolutivo.
    - RUP (Rational Unified Process) para el desarrollo orientado a objetos.
    - Modelos agiles como XP (Extreme Programming), scrum, kanban, scrumban, etc.
  - Todos estos modelos definen una forma ordenada y sistemática para el desarrollo de un proyecto de software.

- En la mayoría de los casos, todos los modelos ofrecen fases o pasos de diseño bien definidos.
- Sin embargo, estas fases deben ser completadas con **un resultado en forma de un documento.** La fase de terminación se conoce comúnmente como ***milestone***.
- **Milestone se logra cuando los documentos requeridos son completados y se ajustan a los criterios de calidad dados.**
- Para ello, usualmente en el desarrollo de software se definen roles dedicados a tareas especificas.
- Incluso algunos modelos definen las técnicas y procesos que deben ser usados en ciertas fases.
- Con ayuda de estos modelos es posible realizar una planeación detallada del uso de recursos.
  - Tiempo, personal, infraestructura, etc.

- En un Proyecto de software, los modelos de desarrollo definen las tareas colectivas y obligatorias, así como su secuencia cronológica.

- **Las pruebas aparecen en cada uno de estos modelos del ciclo de vida del software, pero con significados muy diferentes y en un grado diferente.**

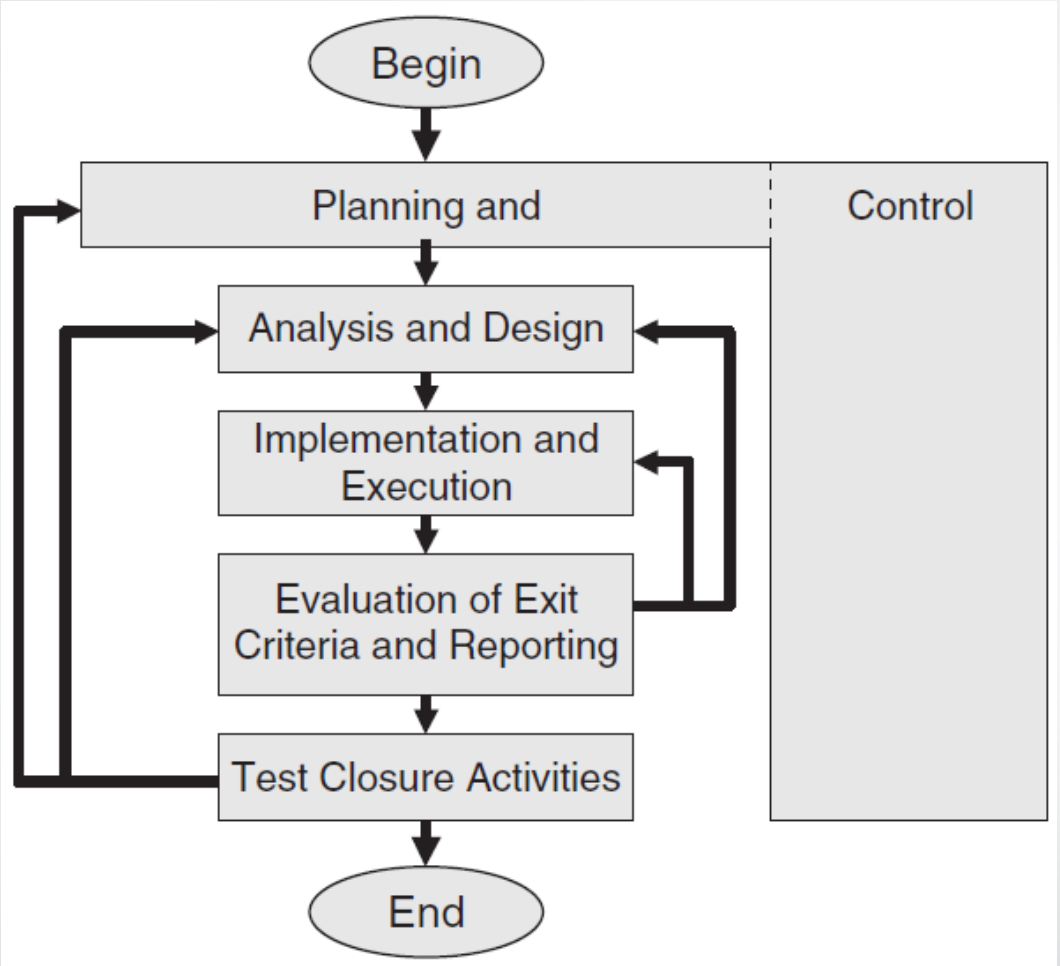
- Por ejemplo, el modelo de cascada es el modelo más básico y fácil de comprender.
  - Sólo cuando una fase de desarrollo es completada se puede pasara a la siguiente fase.
  - Sólo entre dos fases adjuntas puede existir ciclos de retroalimentación.
  - La principal desventaja de este modelo es que la fase de pruebas es entendida como una actividad que se ejecuta una sola ves, al final de proyecto. Antes de que el software sea liberado.
  - **Las pruebas son vistas como una "inspección final".**



¿y si los errores están en la etapa de requerimientos, análisis o diseño?

- La descripción de las tareas en este modelo no son suficientes para las pruebas de un proyecto de software.
- Es necesario incluir las pruebas en todo el proceso de desarrollo.
- Se requiere de un proceso más detallado de las tareas relacionadas a las pruebas.

- La tarea de evaluar (testing) debe ser dividido en subtareas más pequeñas como:
  - Planeación y control de pruebas.
  - Análisis y diseño de pruebas.
  - Implementación y ejecución de las pruebas.
  - Evaluación de las pruebas basado en los criterios de salida e informes de las mismas.
  - Cierre de la actividad de pruebas.



Modelo general del proceso de pruebas.

- Las actividades del proceso de pruebas pueden desarrollarse de manera concurrente.
- Las actividades del proceso de pruebas deben ajustarse a las necesidades de cada proyecto.

Planeación y control de pruebas

- La ejecución de una tarea tan importante como las pruebas no debe realizarse sin un plan.
- La planeación del proceso de pruebas inicia desde el principio del proyecto de software.
- Como en toda planeación, durante el curso del proyecto los planes iniciales deben ser regularmente revisados, actualizados y ajustados.
- El objetivo de las pruebas debe ser definido, así como los recursos necesarios para el proceso de pruebas.
  - ¿Qué empleados se requieren para la ejecución de qué tareas y cuándo?
  - ¿Cuánto tiempo será necesario?
  - ¿Qué equipo e infraestructura debe estar disponible para la ejecución de las pruebas?
  - Estas y otras preguntas deben realizarse durante la planeación.

El resultado de la planeación de pruebas debe ser un documento.

Planeación y control de pruebas

- El **control** es el monitoreo de las actividades de las pruebas y la comparación de lo que está actualmente sucediendo con el proyecto con respecto al plan.
  - Incluye reportar el estado de las desviaciones con respecto al plan y las acciones tomadas para alcanzar los objetivos planeados en base a la nueva situación.
  - Actualizar el plan de pruebas en base a las nuevas situaciones.
- Parte de las tareas en la gestión de pruebas es:
  - Administrar y mantener el proceso de pruebas.
  - La infraestructura de las pruebas.
  - El software de pruebas.

Estrategias de pruebas

- La principal tarea de la planeación es determinar la estrategia de pruebas o enfoque a seguir.
  - Dado que no es factible el uso de pruebas exhaustivas, las prioridades deben basarse en las evaluaciones de riesgo.
  - Las actividades de pruebas deben ser distribuidas a los subsistemas individuales, dependiendo del riesgo esperado y la gravedad de los efectos de las fallas.
  - Se debe dar mayor atención a los subsistemas críticos, realizando pruebas con mayor intensidad.
  - Pruebas menos intensas pueden ser suficientes para aplicar a los subsistemas menos críticos.
  - Si los efectos esperados no son negativos cuando las fallas se presentan, algunas pruebas podrían ser omitidas. Sin embargo, esta decisión debe realizarse con mucho cuidado.

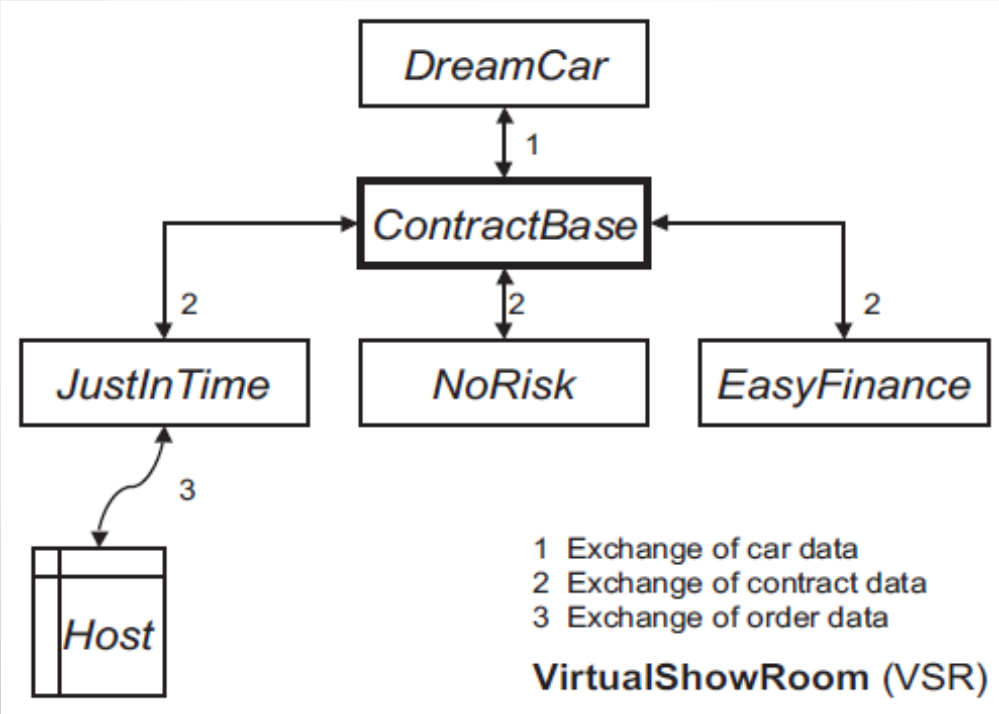
- El **objetivo** de las estrategias de prueba es la **distribución optima de las pruebas** dirigidas a las partes apropiadas del sistema bajo prueba.

Caso de estudio

- **VirtualShowRoom (VSR).** Es un nuevo producto para ventas electrónicas que desea utilizar una empresa dedicada a la fabricación de vehículos.
  - Este software será instalado en todas las concesionarias ubicadas alrededor del mundo.
  - Los clientes interesados en comprar un nuevo carro serán capaces de configurar el modelo de carro deseado (modelo, tipo, color, equipamiento, etc.) con o sin la ayuda de un agente de ventas.
  - El sistema muestra posibles modelos y combinaciones de equipamiento adicional.
  - El sistema debe mostrar de manera instantánea el precio del carro configurado por el cliente. Un subsistema llamado ***DreamCar*** ofrecerá esta funcionalidad.
  - El cliente podrá determinar el plan de financiamiento más apropiado (***EasyFinace***), así como realizar su pedido en línea (***JustInTime***).
  - El cliente deberá tener la capacidad de elegir la aseguradora (***NoRisk***).
  - Tanto la información personal del cliente como la información del contrato de compra-venta será gestionada por el subsistema ***ContractBase.***

Caso de estudio

- Considera que cada subsistema será diseñado y desarrollado por un equipo diferente de desarrollo.



Arquitectura general del sistema.

Ejemplo para una estrategia de pruebas

- El sistema **VSR** consiste de los siguientes subsistemas:
  - ***DreamCar*** permite la configuración individual de un carro y su equipamiento adicional.
  - ***ContractBase*** administra la información del cliente y los datos del contrato.
  - ***JustInTime*** implementa la capacidad de colocar los pedidos en línea.
  - ***EasyFinance*** calcula el método optimo de financiamiento para el cliente.
  - ***NoRisk*** ofrece la capacidad de comprar un seguro.

¿Los 5 subsistemas deben ser probados con la misma intensidad?

Ejemplo para una estrategia de pruebas

- Como resultado de una discusión con el cliente del sistema VSR se establece que un comportamiento incorrecto de los subsistemas *DreamCar* y *ContractBase* puede provocar los efectos más grabes.

- En este caso la estrategia de pruebas dedicada a estos dos subsistemas debe ser realizada con mayor intensidad que en los otros subsistemas.

- Se establece que las ordenes en line colocadas por el subsistema *JustInTime* es menos critica, porque en el peor de los casos, éstas pueden ser enviadas por otros medios (por ejemplo fax). Pero es importante que los datos de la orden no sean alterados o se perderán en el subsistema *JustInTime*.
  - Esto permite definir que se debe realizar un mayor esfuerzo al momento de probar esta propiedad.

Ejemplo para una estrategia de pruebas

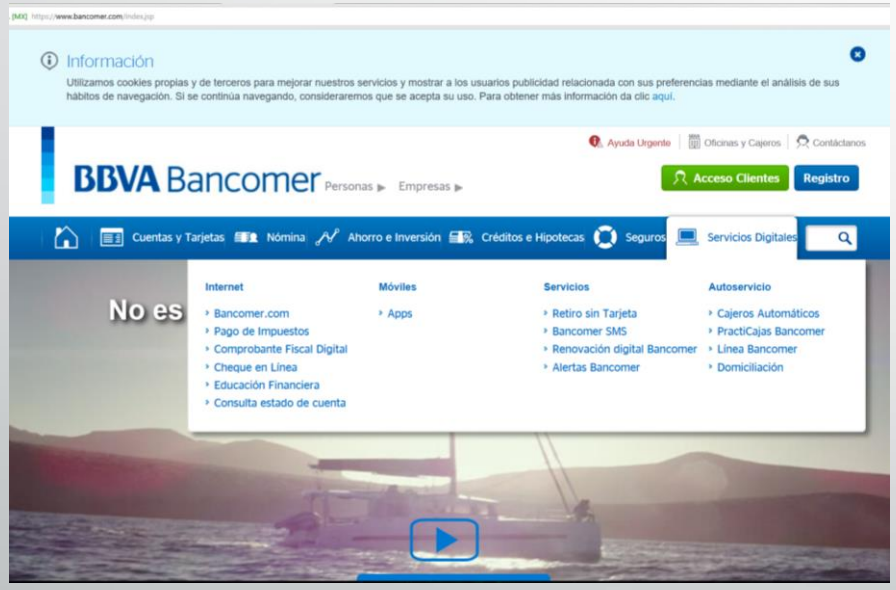
- Para los otros dos subsistemas (*NoRisk* y *EasyFinace*), la estrategia de pruebas definida establece que todas sus funciones principales (calcular tarifas, colocar contratos, guardar e imprimir contratos, etc.) deben ser probadas.

- Debido a las restricciones de tiempo, no es posible cubrir todas las posibles variaciones de contratos para financiar y asegurar un carro. Por lo tanto, se decide realizar pruebas alrededor de las combinaciones y casos con mayor probabilidad o frecuencia (casos más comunes).

Esta puede ser una primera aproximación para definir la estrategia de pruebas del sistema, que busca establecer el nivel de intensidad de las pruebas dentro de todo el sistema.

Ejercicio

- Determinar los riesgos y clasificarlas como bajo, medio y alto.
- Proponer el nivel de intensidad de las pruebas que se deberían realizar en lo siguientes sistemas.



Tarea 3. Participación en foro

- Realizar una reunión con sus asesores (director de tesis y codirector) y en caso de existir un cliente, involúcralo para identificar los riegos del sistema a desarrollar.
- Determinar los riesgos y clasificarlas como bajo, medio y alto.
- Proponer el nivel de intensidad de las pruebas que se deberían realizar en los subsistemas o módulos del sistema a desarrollar.
- Participa en el foro y comparte la estrategia del plan de pruebas generado en base a esta actividad.
- Ofrece una retroalimentación por lo menos a uno de tus compañeros.

Análisis y diseño de las pruebas

1) *Revisar la base de las pruebas.*

- La primera tarea es revisar las especificaciones de lo qué debe ser probado.
- Las especificaciones deben ser concretas y claras para desarrollar los casos de prueba.
- Las bases para los criterios de una prueba pueden ser:
  - La documentación de las especificaciones o la documentación de la arquitectura del sistema.
  - Los resultados del análisis de riesgos.
  - Cualquier otro documento creado durante el proceso del desarrollo del sistema.

Análisis y diseño de las pruebas

- Ejemplo:
  - Un requerimiento puede ser demasiado impreciso al definir el resultado esperado o la funcionalidad del sistema. En este contexto no es posible desarrollar un caso de prueba apropiado.
  - La testabilidad (testability) sería insuficiente.
- Las precondiciones y requerimientos para el diseño del caso de prueba deben ser determinadas en base a un análisis de los requerimientos, la funcionalidad esperada y la estructura de los objetos de prueba.

Análisis y diseño de las pruebas

2) *Revisar la testabilidad (testability)*

- Este proceso incluye comprobar la factibilidad con la que se pueden abordar las interfaces (apertura de interfaz) y la facilidad con la que el objeto de prueba (test object) se puede separar en unidades más pequeñas, más fáciles de comprobar.

- El resultado de este análisis también puede ser utilizado para priorizar las condiciones de prueba basado en los objetivos generales de las pruebas.

Análisis y diseño de las pruebas

3) *Revisar el riesgo*

- La estrategia de pruebas determinada en el plan de pruebas define la técnica de pruebas que deberá ser utilizada.

- La estrategia de prueba depende de los requisitos de fiabilidad y seguridad. Si hay un alto riesgo de falla para el software, se deben planificar pruebas muy minuciosas. Si el software es menos crítico, la prueba puede ser menos formal.

Análisis y diseño de las pruebas

4) Casos de prueba lógicos y concretos

- La especificación de los casos de prueba consiste en dos pasos:
  - **Casos de prueba lógicos.** Consiste en la definición lógica de los casos de prueba. La base de pruebas guía la selección de los casos de prueba lógicos. Los casos pueden ser determinados a partir de las especificaciones de los objetos de prueba.
  - **Casos de prueba concretos.** Esto significa que las entradas para los casos de prueba han sido seleccionadas.

Análisis y diseño de las pruebas

5) *Test oracle*

- **Es el mecanismos para predecir los resultados esperados.** El tester debe obtener información de fuentes apropiadas (documentación).

- Las especificaciones pueden servir como un test oracle.

Análisis y diseño de las pruebas

6) *Casos de prueba para entradas esperadas y no esperadas*

- Los casos de prueba pueden ser basados en dos criterios:
  - **Casos de prueba orientados a examinar la salida y funcionalidad esperada.** Se incluyen los casos de prueba orientados a las excepciones y errores especificados. Estos casos son conocidos como casos de prueba negativos. Ejemplo, sobrecarga en la capacidad de conectividad.
  - **Casos de prueba para examinar la reacciones de los objetos de prueba** para validar entradas inesperadas o condiciones, las cuales no han sido especificadas dentro del manejo de excepciones.

Ejemplo

El siguiente ejemplo intenta clarificar la diferencia entre los casos de prueba lógicos y concretos.

- Usando el software de ventas, el distribuidor de automóviles es capaz de definir reglas de descuentos para sus agentes de ventas:
  - Vehículos con un precio **menor a** 15,000 dls no tienen descuento.
  - Vehículos con un precio **mayor o igual a** 15,000 **y hasta** 20,000 dls tendrán un descuento del 5%.
  - Vehículos con un precio **mayor a** 20,000 **pero menor a** 25,000 tendrán un descuento de 7%.
  - Vehículos con **un precio de** 25,000 **o superior** tendrán un descuento del 8.5%.

- A partir de esta información se pueden proponer los siguientes casos:
  - Precio < 15000, descuento = 0%
  - 15000 ≤ Precio ≤ 20000, descuento = 5%
  - 20000 < Precio < 25000, descuento = 7%
  - Precio ≥ 25000, descuento = 8.5%

Ejemplo

Tabla con casos de prueba lógicos.

Casos de prueba lógicos	1	2	3	4
Valor de entrada x (precio en dolares)	$x < 15000$	$15000 \leq x \leq 20000$	$20000 < x < 25000$	$x \geq 25000$
Resultado esperado (descuento en %)	0	5	7	8.5

Tabla con casos de prueba concretos.

Casos de prueba concretos	1	2	3	4
Valor de entrada x (precio en dolares)	14500	16500	24750	31800
Resultado esperado (descuento)	0	825	1732.50	2703

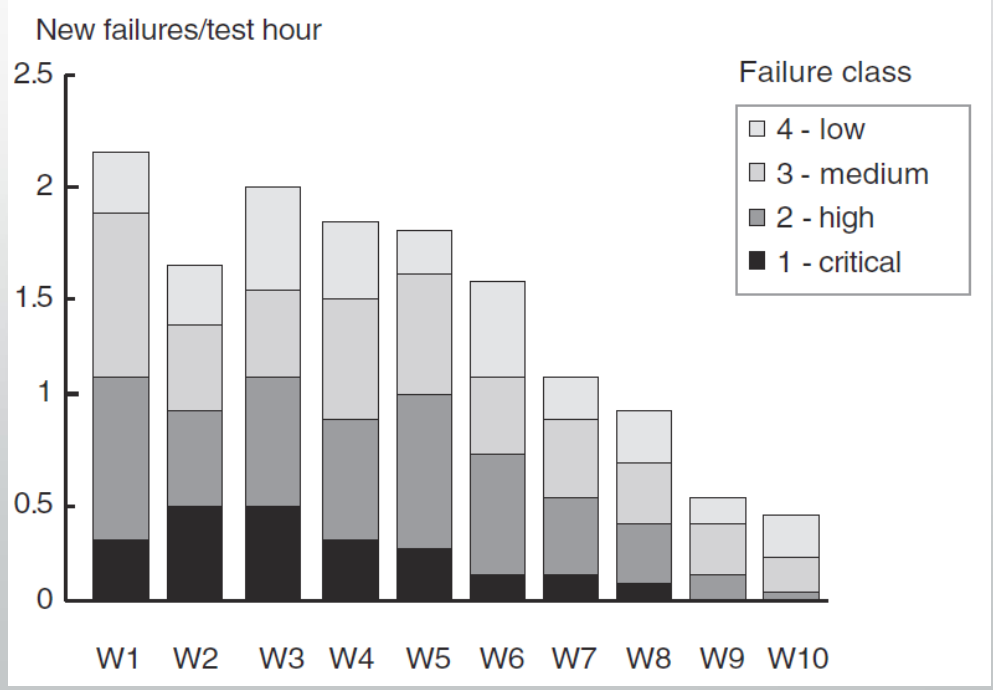
Implementación y ejecución de pruebas

- En esta fase todos los casos de prueba lógicos deben ser transformados a casos de prueba concretos.
- Todos los detalles del entorno para el desarrollo de pruebas deben ser configurados.
- Las pruebas deben ser ejecutadas y registradas.
- Es importante definir la forma en que las pruebas serán ejecutadas.
  - Prioridad de las pruebas.
  - Si el desarrollador de las pruebas será quien las ejecute o debe ser alguien más.
  - Para una ejecución eficiente y un mejor entendimiento, los casos de prueba deben ser agrupados en repositorio de pruebas o escenario de pruebas.
- Los fallos que se encuentre durante la ejecución de las pruebas deberán ser documentados.
- Es importante asegurar que las fallas encontradas no sean consecuencia de una ejecución errónea de las pruebas.

Evaluación de pruebas e informe

- En esta fase, el objeto de prueba se evalúa en función a los criterios de salida de las prueba. Estos criterios fueron especificados durante la planeación. El resultado de este proceso podría ser:
  - Una terminación normal de las pruebas si todos los criterios son alcanzados.
  - Considerar el ejecutar pruebas adicionales.
  - Los criterios fueron muy difíciles de alcanzar.

- En adición a los criterios de cobertura de las pruebas. Otros criterios pueden ser utilizados para definir el final de las pruebas.
  - Un posible criterio puede ser la tasa/frecuencia de las fallas.
  - Si el número de fallas es menor al umbral definido, se puede decir que económicamente no es justificable continuar con las pruebas.



Evaluación de pruebas e informe

- Las fallas encontradas durante las pruebas deben ser reparadas. Después de la corrección se deben ejecutar las pruebas agregando nuevas pruebas.

- Si se detectan nuevas fallas, este proceso cíclico debe ser ejecutado nuevamente.

- Cundo los criterios de prueba son totalmente alcanzados o por lo menos el porcentaje deseado. Se debe escribir un reporte de las pruebas para las partes interesadas en el proyecto.
  - Administrador del proyecto.
  - Administrador de las pruebas.
  - Posiblemente el cliente.

Cierre de las actividades de prueba

- El conocimiento adquirido durante el desarrollo de las pruebas debería ser analizado y puesto a disposición para futuros proyectos.

- Información importante para una evaluación puede ser extraída partiendo de las siguientes preguntas:
  - ¿Cuáles de los resultados planificados fueron alcanzados y cuándo?
  - ¿Qué eventos inesperados sucedieron?
  - ¿Existe algún problema abierto (sin resolver) y solicitudes de modificaciones?
    - ¿Por qué no fueron implementadas?
  - ¿Cómo fue la aceptación del usuario después de implementar el sistema?

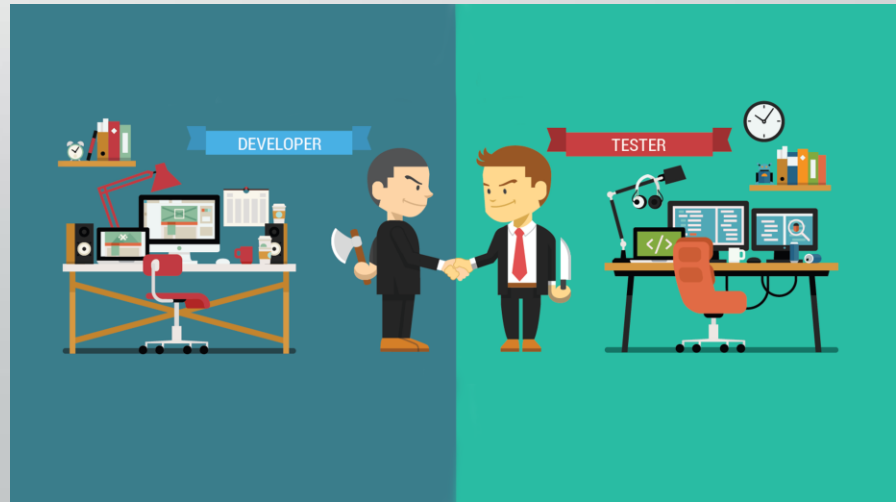
Cierre de las actividades de prueba

- Otra actividad del cierre de las pruebas, es conservar el software de pruebas (the testware) para un uso futuro.
  - Durante el tiempo de vida del sistema se pueden encontrar nuevas fallas que las pruebas no pudieron detectar.
  - Los clientes pueden solicitar modificaciones.

- Gran parte del esfuerzo requerido durante el mantenimiento del sistema puede ser evitado si se conservan las pruebas.
  - Casos de pruebas.
  - Registros de las pruebas.
  - Infraestructura.
  - Herramientas.
  - Etc.

La psicología de las pruebas

- Uno de los objetivos de las pruebas de software es encontrar discrepancias entre el software y las especificaciones o las necesidades del cliente.
- Las tareas del desarrollador de software son vistas generalmente como acciones constructivas.
- Las tareas del encargado de revisar la documentación y el software son vistas comúnmente como tareas destructivas.
- ¿Puede un programador probar su propio software?
  - Esta es una pregunta que se realiza frecuentemente.
  - No existe una respuesta universal valida.



La psicología de las pruebas

- El principal problema de que los desarrolladores realicen la fase de pruebas de sus propios sistemas es que son muy optimistas.

- Existe el peligro de que los desarrolladores olviden el objetivo de las pruebas.
  - Frecuentemente los desarrolladores se interesan más en programar que en probar.
  - Es común que realicen pruebas superficiales.
  - ¿A quién le gusta detectar y mostrar sus propios errores?

- Por otra parte, si un desarrollador participa en las pruebas sería ventajoso tener un conocimiento preciso de nuestros objetos de prueba.
  - Ahorro de tiempo al no existir la necesidad de estudiar el objeto de prueba.

Principios generales de las pruebas

Durante los últimos 40 años, 7 principios para las pruebas han sido aceptados como reglas para esta tarea.

1. Las pruebas muestra la presencia de defectos, no su ausencia.
2. Las pruebas exhaustivas son imposibles.
3. Las actividades de prueba deben iniciar tan pronto como sea posible.
4. Agrupamiento de defectos.
5. La paradoja del pesticida.
6. Las pruebas dependen del contexto.
7. La falacia de la ausencia de fallas.

La ética del ingeniero de pruebas

- La participación en las pruebas de software permite a las personas obtener información confidencial y privilegiada.
- Un código de ética es necesario, entre otras razones para asegurar que la información no sea objeto de un **uso inapropiado**.
- el **ISTQB** (International Software Testing Qualifications Board) establece el siguiente **código de ética:**
  - **Público.** Los probadores de software actuarán en coherencia con el interés público.
  - **Cliente y empleador.** Los probadores de software actuarán en el mejor de los intereses de su cliente y empleador, en coherencia con el interés público.
  - **Producto.** Los probadores de software garantizarán que los productos entregables que proporcionan (por lo que respecta a los productos y sistemas que prueban) se ajustan a los más altos estándares profesionales.

La ética del ingeniero de pruebas

- **Criterio.** Los probadores de software mantendrán la integridad y la independencia en su criterio profesional.
- **Gestión.** Los jefes y líderes de pruebas de software suscribirán y fomentarán un enfoque ético en la gestión de las pruebas de software.
- **Profesión.** Los probadores de software aumentarán la integridad y la reputación de la profesión en coherencia con el interés público.
- **Compañeros.** Los probadores de software serán equitativos y apoyarán a sus compañeros fomentando la cooperación con los desarrolladores de software.
- **Nivel individual.** Los probadores de software participarán en un aprendizaje a lo largo de toda la vida por lo que respecta a la práctica de su profesión y fomentarán la adopción de un enfoque ético en la práctica de su profesión.