
0 – Introducción

01 – Presentación

Organización del curso

- Agenda del curso.
- Por favor mantenga su celular apagado.

0 - Introducción

02 - Tabla de contenidos

El presente curso se ha desarrollado de acuerdo con el programa de estudios del año 2010 de Probador, Certificado Nivel Básico, que consta de siete capítulos:

Capítulo 0 **Introducción.**

Capítulo I Fundamentos de Pruebas.

Capítulo II Pruebas a través del Ciclo de Vida Software.

Capítulo III Técnicas Estáticas.

Capítulo IV Técnicas de Diseño de Pruebas.

Capítulo V Gestión de Pruebas.

Capítulo VI Herramientas de Pruebas.

0 - Introducción

03 - Organizaciones internacionales

Programa de capacitación del ISTQB*

- En 1998. Se desarrolla en Gran Bretaña, un programa de capacitación de múltiples niveles.
- Los fundamentos del proceso de pruebas software son formulados en el programa de estudios para el nivel básico (Syllabus for Foundation Level - edición actual: marzo de 2010).
- Desde 2004 también se cuenta con certificaciones para el Nivel Avanzado (Advanced Level): Jefe de Pruebas (Test Manager). Probador Técnico (Technical Tester). Probador Funcional (Functional Tester)
- El Nivel Experto se encuentra listo para descarga.
- Los Comités de Pruebas (Testing Boards) locales de cada país conforman la estructura de la organización del "International Software Testing Qualification Board" (www.istqb.org)
- Por ejemplo en Hispanoamérica: HISPANIC America Software Testing Qualification Board (HASTQB)

*ISTQB = International Software Testing Qualification Board (Comité Internacional de Clasificación de pruebas de software)

0 - Introducción

04 - Programa de estudios y evaluación

**El conjunto de diapositivas está basado en el
Programa de Estudios de
Probador Certificado - Nivel Básico del ISTQB
Versión 2010 (marzo).**

- A continuación del curso de formación tendrá lugar un examen al que se puede asistir para obtener el certificado de Probador Certificado. Nivel Básico (Certified Tester Foundation Level).
- La evaluación es realizada por un examinador perteneciente a una organización independiente (por ejemplo ISQI*).
- Los temas de la evaluación están extraídos de las secciones correspondientes a las dadas en el curso de formación. El examen es del tipo selección múltiple, con una duración de 60 minutos.
- Cada pregunta tiene una (de cuatro) única respuesta correcta. Tienen que ser respondidas un total de 40 preguntas, de las cuales 26 (65%) tienen que ser respondidas de forma correcta con el objeto de aprobar el examen

* ISQI International Software Quality Institute

0 - Introducción

05 - Objetivos

Objetivos principales para la formación de Probador

Certificado

- Aprender las técnicas básicas para planificar las pruebas.
- Aplicar las técnicas de pruebas software en proyectos.
- Seleccionar las técnicas y objetivos de pruebas apropiados.
- Aprender una terminología común.

Audiencia

- El curso está dirigido a probadores de software, desarrolladores y jefes de proyecto en un entorno de producción software así como en un entorno de producción industrial que deseen dar a su conocimiento un fundamento de mayor solidez.

Documentos

- Conjunto de diapositivas propios de la presentación.
- Ejercicios y sus soluciones.
- Referencias y glosario.

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- **I/01 ¿Por qué es necesario Probar?**
- I/02 ¿Qué son pruebas?
- I/03 Los siete Principios generales del proceso de pruebas de software.
- I/04 Proceso de pruebas básico.
- I/05 Psicología en el proceso de pruebas.
- I/06 Código de Ética.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

La importancia económica del software

- El funcionamiento de maquinaria y equipamiento depende en gran medida del software.
- No es posible imaginar grandes sistemas, en el ámbito de las finanzas ni el control de gráfico, funcionando sin software.

Calidad software

- Cada vez más, la calidad software se ha convertido, en un factor determinante del éxito técnico o comercial de sistemas y producidos.

Pruebas para la mejora de la calidad

- Las pruebas y revisiones aseguran la mejora de la calidad de productos software así como de la calidad del proceso de desarrollo en sí.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Causas de los fallos (failures) de software

- **Error humano**

Un defecto es introducido en el código software, en los datos o en los parámetros de configuración.

Causas del error humano:

Plazos, demandas excesivas debidas a lo complejidad, distracciones.

- **Condiciones ambientales**

Cambios en las condiciones ambientales.

Causas de condiciones ambientales negativas/adversas.

Radiación, magnetismo, campos electromagnéticos y polución, manchas solares, fallo de discos duros, fluctuaciones en el suministro de energía eléctrica.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Error, defecto (defect), fallo (failure)

- **Error (IEEE 610):**

Acción humana que produce un resultado incorrecto por ej. Un error de programación.

- **Defecto (Defect):**

Desperfecto en un componente o sistema que puede ser una causa por la cual el sistema o componente no logre llevar o cabo su función específica, por ej. Sentencia o definición de datos incorrectas.

- **Fallo (Failure):**

Manifestación física o funcional de un defecto. Si un defecto es encontrado durante la ejecución de una aplicación puede producir un fallo.

Desviación de un componente o sistema respecto de la prestación, el servicio o resultado esperados.

UN ERROR INTRODUCE UN DEFECTO, UN DEFECTO CAUSA UN FALLO.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Proceso de pruebas durante el proceso de desarrollo, mantenimiento y operaciones

- **Mejora de la calidad de un producto software:**

El proceso de pruebas ayuda a proveer al software de los atributos deseados. Puede retirar defectos que conducen a fallos.

- **Reducción del riesgo de detectar errores:**

Actividades de pruebas software adecuadas reducirán el riesgo de encontrar errores durante la fase de operaciones software.

- **Satisfacer compromisos:**

La ejecución de pruebas puede ser un requisito obligatorio por parte del cliente o debido a normas legales así como el cumplimiento de estándares propios de la Industria específica.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

El costo de los defectos

- El costo de eliminar defectos se incrementa con el tiempo durante el cual el defecto permanece en el sistema
- La selección de errores en etapas tempranas permite la corrección de los mismos a costos reducidos.



I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Calidad software

- **Definición: Software (según IEEE 610):**
Programas de computadora, procedimientos, y posiblemente documentación asociada así como los datos relacionados con la operación de un sistema Informático.
- **Definición: Calidad software (según ISO/IEC 9126):**
La totalidad de la funcionalidad y características de un producto software que contribuyen a su habilidad de satisfacer necesidades especificadas o implícitas
- **Definición: Calidad (según IEEE Std 610):**
Grado en el cual un componente, sistema o proceso satisface los requisitos especificados y/o necesidades del usuario/cliente y sus expectativas.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Calidad software

- **De acuerdo a la norma ISO/IEC 9126 la calidad software está constituida por:**
 - Funcionalidad Atributos funcionales de calidad
 - Fiabilidad }
 - Usabilidad }
 - Eficiencia }
 - Mantenibilidad }
 - Portabilidad }

Atributos no-funcionales de calidad
- **Tipos de Aseguramiento de la Calidad (QA):**
 - **Actividades constructivas** con el objeto de prevenir defectos, por ejemplo a través de la aplicación de métodos apropiados de ingeniería de software.
 - **Actividades analíticas** con el objeto de prevenir defectos. por ejemplo a través de pruebas que conducen a la corrección de defectos y prevención de fallos, incrementando así la calidad del software.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Aseguramiento de la calidad constructivo

Proceso de calidad - Gestión de la calidad.

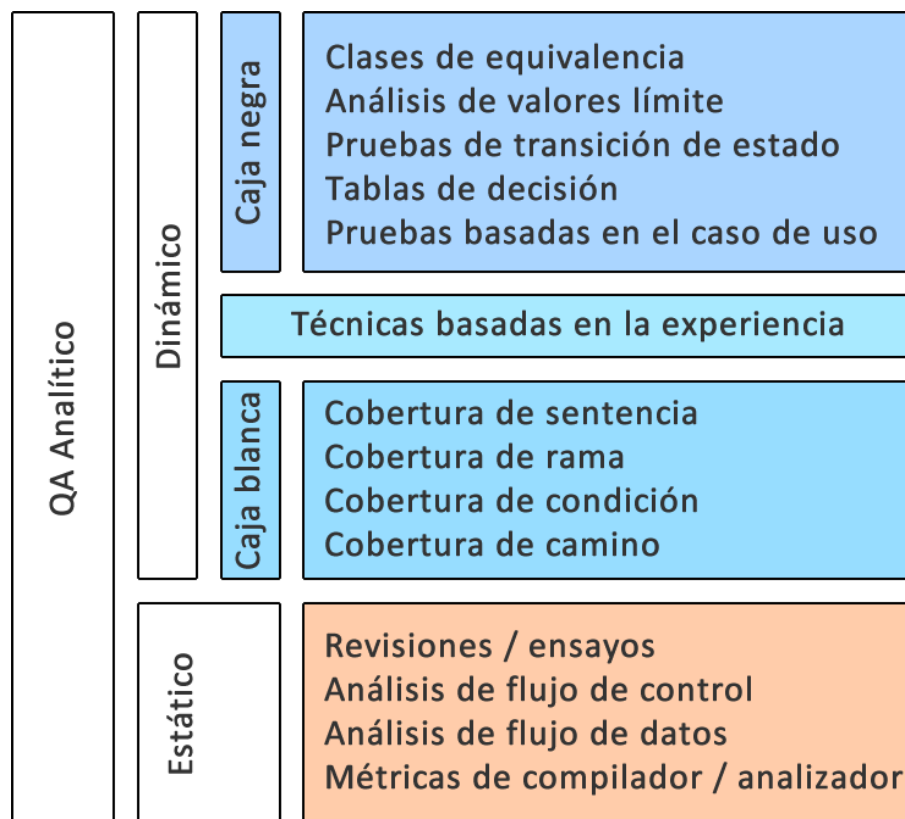
QA Constructivo	Organizacional	Guías Estándares Listas de Comprobación Reglas de proceso y normas Requisitos legales	Consigna Los defectos evitados no requieren ser reparados. Los errores cometidos en el pasado no deben ser repetidos.
	Técnico	Métodos Herramientas Lenguajes Listas / plantillas Entorno de Desarrollo Integrado (IDE)	Prevenir defectos

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Aseguramiento de la calidad analítico

Calidad de producto - Procedimientos de Verificación y Pruebas



Consigna

Los defectos deben ser detectados tan temprano en el proceso como sea posible.

Pruebas estáticas examen sin la ejecución del programa.

Pruebas dinámicas incluye la ejecución del programa.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Calidad software - Atributos funcionales de calidad

Funcionalidad significa:

- **Correcto:** La funcionalidad satisface los atributos / capacidades requeridos.
- **Compleitud:** La funcionalidad satisface todos los requisitos (funcionales).

Funcionalidad incluye (según ISO/IEC 9126):

- Adecuación.
- Precisión.
- Conformidad (Estándares).
- Interoperabilidad (Componentes).
- Seguridad.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Calidad software - Atributos no funcionales de calidad / 1

Fiabilidad

- Madurez, tolerancia a defectos, recuperación tras fallos.
- Características: En determinadas condiciones, el software / sistema mantendrá su capacidad / funcionalidad a lo largo de un período de tiempo.
- Fiabilidad = Calidad / tiempo.

Usabilidad

- Facilidad de ser utilizado, facilidad de ser aprendido, facilidad de ser comprendido, atractivo.
- Características: fácil de usar, fácil de aprender, conforme o normas, uso intuitivo.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Calidad software - Atributos no funcionales de calidad /2

Eficiencia

- Comportamiento del sistema: funcionalidad y respuesta temporal.
- Características: El sistema requiere la utilización de un mínimo de recursos (por ejemplo tiempo de CPU) para ejecutar una tarea determinada.

Mantenibilidad

- Verificable, estable, cualidad de ser analizado, modificable.
- Características: Medida del esfuerzo requerido para realizar cambios en un sistema / componentes.

Portabilidad

- Capacidad de ser reemplazado, Instalable.
- Capacidad del software de ser transferido a un nuevo entorno (software, hardware, organización).
- Características: Fácil de instalar y desinstalar, parámetros.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Atributos de la calidad

- Algunos atributos de la calidad de un producto software se influyen mutuamente. Debido a esta interdependencia y en función del objeto de prueba, los atributos deberán ser caracterizados por una prioridad a los efectos de ser tenida en cuenta. Por ejemplo eficiencia vs. portabilidad.
- Se realizarán distintos tipos de pruebas con el objeto de medir cada tipo de atributo.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Objetivos de las pruebas

Adquirir conocimiento sobre los defectos en un objeto de prueba

- Los defectos propios de un objeto de prueba deben ser detectados y descritos de tal forma que se facilite su corrección.

Comprobar la funcionalidad

- La funcionalidad del sistema debe ser implementada tal y como ha sido especificada.

Generar información

- Se debe proporcionar Información relativa a evitar riesgos relativos a un sistema software antes de su entrega a los usuarios. La adquisición de esta Información puede ser uno de los objetivos de las pruebas.

Generar confianza

- Un sistema software que ha sido probado de forma adecuada se considera que cumple con la funcionalidad esperada y cuenta con un alto nivel de calidad.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

¿Cuántas pruebas son suficientes?

Criterio para la finalización de pruebas

No encontrar (más) defectos es un criterio apropiado para finalizar las actividades de pruebas. Sin embargo son necesarias otras métricas para reflejar de forma adecuada el nivel de calidad alcanzado.

- **Pruebas basadas en el riesgo**

- El nivel de riesgo determina el grado en el cual se ha probado, es decir: responsabilidad en caso de fallos, probabilidad de la ocurrencia de fallos, aspectos relativos o factores económicos y propios del proyecto.

- **Pruebas basadas en plazos y presupuesto**

- La disponibilidad de recursos (personal, tiempo, presupuesto) puede determinar la medida del esfuerzo a dedicar al proceso de pruebas.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Caso de prueba (test case), base de una prueba (test basis)

- **Caso de prueba (test case):**

La definición de un caso de prueba incluye la siguiente información (según IEEE Std 610):

- Precondiciones.
- Conjunto de valores de entrada.
- Conjunto de resultados esperados.
- Forma en la cual se debe ejecutar el caso de prueba y verificar los resultados.
- Pos condiciones esperadas.

- **Base de la prueba (test basis o test base):**

Conjunto de documentos que definen los requisitos de un componente o sistema. Utilizado como fundamento para el desarrollo de casos de prueba.

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Desarrollo de software y revisiones:

- **Código (code), código fuente (source code):**
Programa de computadora escrito en un lenguaje de programación que puede ser leído por una persona.
- **Depuración (debugging):**
Localización y corrección de defectos en el código fuente
- **Desarrollo software (software development):**
Es un proceso / secuencia de actividades cuyo objetivo es desarrollar un sistema basado en un computador (ordenador). Normalmente sigue un modelo de desarrollo software.
- **Requisito (requirement):**
Un requisito describe un atributo funcional deseado o considerado obligatorio.
- **Revisión (review):**
Evaluación de un producto o estado de un proyecto con el objeto de detectar discrepancias con respecto a los resultados esperados (planificados) y para recomendar mejoras (según IEEE Std 1028).

I - Fundamentos de Pruebas Software

01 - ¿Por qué es necesario Probar?

Resumen:

- Los fallos de software pueden causar importantes perjuicios.
- La calidad software es la suma de los atributos que se refieren a la capacidad del software de satisfacer un conjunto de requisitos dados.
- El aseguramiento de la calidad constructivo se ocupa de la prevención de defectos.
- El aseguramiento de la calidad analítico se ocupa de detectar y corregir defectos.
- Los atributos de la calidad funcionales y no funcionales definen la calidad total del sistema.
- Cada prueba debe contar con un criterio para la finalización de pruebas. Al alcanzar el criterio de finalización de pruebas finalizan las actividades del proceso de pruebas.
- Los probadores (testers) buscan fallos en el sistema e informan sobre los mismos (proceso de pruebas). Los desarrolladores buscan defectos y los corrigen (depuración).

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- I/01 ¿Por qué es necesario Probar?
- **I/02 ¿Qué son pruebas?**
- I/03 Los siete Principios generales del proceso de pruebas de software.
- I/04 Proceso de pruebas básico.
- I/05 Psicología en el proceso de pruebas.
- I/06 Código de Ética

I - Fundamentos de Pruebas Software

02 - ¿Qué son pruebas?

Otros medios de prueba más de la ejecución de pruebas

- Ejecución de las pruebas es sólo una parte de las pruebas
- El proceso de prueba incluye:
 - Planificación y control
 - Elección de las condiciones de prueba
 - Diseño y ejecución de casos de prueba
 - Control de resultados
 - Evaluación de los criterios de salida
 - Presentación de informes sobre el proceso de prueba y el sistema bajo prueba
 - Finalizar o completar las actividades de cierre
- Revisión de documentos, códigos fuente y la realización de análisis estático también ayudan a prevenir los defectos que aparecen en el código.

I - Fundamentos de Pruebas Software

02 - ¿Qué son pruebas?

Objetivos de pruebas

- **Búsqueda de defectos**

Puede ser:

- En las pruebas de desarrollo: para encontrar el mayor número posible de errores
- En las pruebas de aceptación: para confirmar que el sistema funciona como se esperaba

- **Ganar confianza en el nivel de calidad**

- **Proporcionar información para la toma de decisiones**

Para evaluar la calidad del software, para dar información a los interesados de los riesgos de liberar el sistema en un momento dado

- **Prevención de defectos**

Las pruebas de mantenimiento permiten determinar si se han introducido nuevos defectos durante el desarrollo de cambios

I - Fundamentos de Pruebas Software

02 - ¿Qué son pruebas?

Término: Desarrollo de software

Depuración

El proceso de encontrar, analizar y eliminar las causas de fallas en el software.

Requisito (requerimiento)

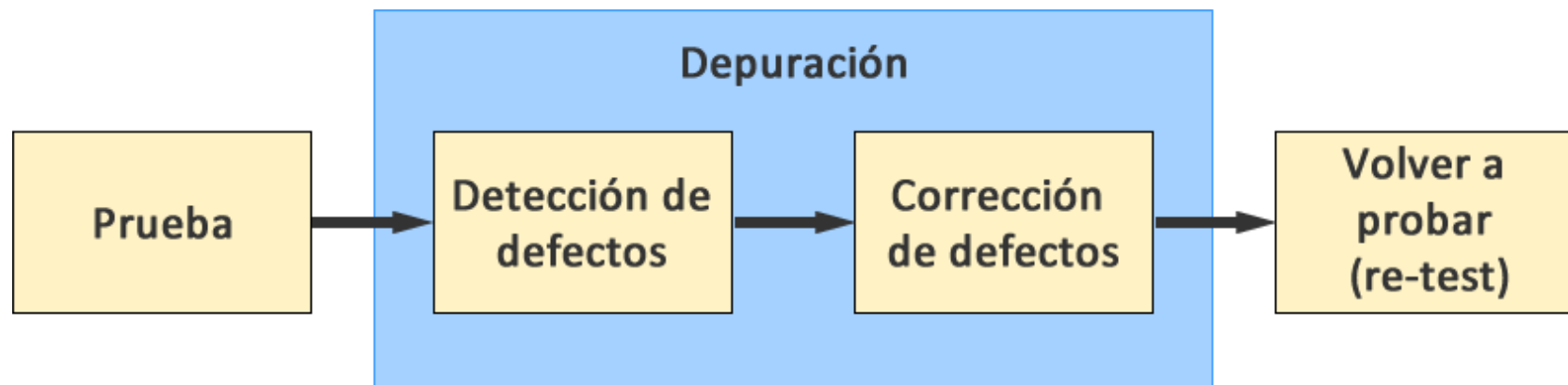
Una condición o capacidad que necesita un usuario para resolver un problema, alcanzar un objetivo que se debe cumplir o que tenga un componente o el sistema, para satisfacer un contrato, norma, especificación u otro documento formalmente impuesto [después de IEEE 610].

Revisión

La evaluación de un estado del producto o proyecto para determinar las discrepancias de los resultados previstos y recomendar mejoras. Las revisiones incluyen [según la IEEE 1028]: revisión de la gestión, revisión informal, revisión técnica, inspección y walkthrough (tutorial).

I - Fundamentos de Pruebas Software
02 - ¿Qué son pruebas?

Pruebas y depuración (debugging)



- **Probar y volver a probar (re-test) son actividades propias del proceso de pruebas.**
Las pruebas muestran los fallos.
Volver a probar (re-test) verifica que el defecto ha sido corregido.
- **La depuración y la corrección de defectos son actividades propias del desarrollo.**
A través de la depuración los desarrolladores pueden reproducir los fallos, analizar el estado del programa y detectar el defecto correspondiente con el objeto de corregirlo.

I - Fundamentos de Pruebas Software

02 - ¿Qué son pruebas?

Resumen

- **El proceso de pruebas fundamentales comprende:**
 - Pruebas de planificación y control.
 - Pruebas de análisis y diseño.
 - Pruebas de implementación y ejecución.
 - Evaluación de criterios de salida y presentación de informes.
 - Actividades de cierre de pruebas.
- **Los objetivos de pruebas pueden ser:** Encontrar defectos, el nivel de calidad, información para la toma de decisiones, la prevención de defectos.
- El diseño de pruebas debe iniciarse en fases tempranas del ciclo de vida del software.
- **Pruebas dinámicas** significa la ejecución del software, las **pruebas estáticas** consiste en la revisión de documentos.
- **Las pruebas dinámicas** muestran fallas que son causadas por defectos, la depuración encuentra, analiza y elimina la causa de la falla.

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- I/01 ¿Por qué es necesario Probar?
- I/02 ¿Qué son pruebas?
- **I/03 Los siete principios generales del proceso de pruebas de software.**
- I/04 Proceso de pruebas básico.
- I/05 Psicología en el proceso de pruebas.
- I/06 Código de Ética

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 1: El proceso de pruebas demuestra la presencia de defectos.

- El proceso de pruebas puede probar la presencia de defectos.
- Las desviaciones identificadas a lo largo del proceso de pruebas demuestran la presencia de un fallo.
- La causa de un fallo puede no ser obvia.
- El proceso de pruebas no puede demostrar la ausencia de defectos.
- Las pruebas reducen la probabilidad de la presencia de defectos que permanecen sin ser detectados. La ausencia de fallos no demuestra la corrección de un producto software.
- El mismo proceso de pruebas puede contener errores.
- Las condiciones de las pruebas pueden ser inapropiadas para detectar errores.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 2: No es posible realizar pruebas exhaustivas

- **Pruebas exhaustivas (exhaustive testing).**
 - Enfoque del proceso de pruebas en el cual el juego de pruebas (suite de pruebas) abarca todas las combinaciones de valores de entrada y precondiciones.
- **Explosión de casos de prueba (test case explosion).**
 - Define el incremento exponencial de esfuerzo y costo en el caso de pruebas exhaustivas.
- **Prueba de muestra (sample test).**
 - La prueba incluye solamente a un subconjunto (generado de forma sistemática o aleatoria) de todos los posibles valores de entrada.
 - En condiciones normales, se utilizan generalmente pruebas de muestra. Probar todas las combinaciones posibles de entradas y precondiciones sólo es económicamente viable en casos triviales.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 3: Pruebas tempranas (early testing)

- La corrección de un defecto es menos costosa en la medida en la cual su detección se realiza en fases más tempranas del proceso software.
- Se obtiene una máxima rentabilidad cuando los errores son corregidos antes de la implementación.
- Los conceptos y especificaciones pueden ser probados.
- Los defectos detectados en la fase de concepción son corregidos con menor esfuerzo y costos.
- La preparación de una prueba también consume tiempo.
- El proceso de pruebas implica más que sólo la ejecución de pruebas.
- Las actividades de pruebas pueden ser preparadas antes de que el desarrollo se haya completado.
- Las actividades de pruebas (incluidos las revisiones) deben ser ejecutadas en paralelo a la especificación y diseño software.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 4: Agrupamiento de defectos (defect clustering)

- Al encontrar un defecto se encontrarán más defectos "cerca"
- Los defectos aparecen agrupados como hongos o cucarachas.
- Cuando se detecta un defecto es conveniente investigar el mismo módulo en el que ha sido detectado.
- Los probadores (testers) deben ser flexibles
- Habiendo sido detectado un defecto es conveniente volver a considerar el rumbo de las pruebas siguientes.
- La identificación de un defecto puede ser investigada con un mayor grado de detalle, por ejemplo, realizando pruebas adicionales o modificando pruebas existentes.

I - Fundamentos de Pruebas Software

03 - Los siete Principios generales del proceso de pruebas de software.

Principio 5: Paradoja del pesticida

- **Repetir pruebas en las mismas condiciones no es efectivo.**
 - Cada caso de prueba debe contar con una **combinación única** de parámetros de entrada para un objeto de pruebas particular, de lo contrario no se podrá obtener información adicional.
 - Si se ejecutan las mismas pruebas de forma reiterada no se podrán encontrar nuevos defectos.
- **Las pruebas deben ser revisadas/modificadas regularmente para los distintos módulos (código).**
 - Es necesario repetir una prueba tras una modificación del código (corrección de defectos, nueva funcionalidad)
 - La automatización de pruebas puede resultar conveniente si un conjunto de casos de prueba se debe ejecutar frecuentemente.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 6: Las pruebas dependen del contexto

- **Las pruebas se desarrollan de forma diferente en diferentes contextos.**
- **Objetos de prueba diferentes son probados de forma diferente.**
 - El controlador del motor de un coche requiere pruebas diferentes respecto de aquellas para una aplicación de "e- Commerce".
- **Entorno de pruebas (cama de pruebas - test bed) vs. entorno de producción.**
 - Las pruebas tienen lugar en un entorno distinto del entorno de producción. El entorno de pruebas debe ser similar al entorno de producción.
 - Siempre habrá diferencias entre el entorno de pruebas y el entorno de producción. Estas diferencias introducen incertidumbre con respecto a las conclusiones que se pudieran obtener tras las pruebas.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Principio 7: La falacia de la ausencia de errores

Un proceso de pruebas adecuado detectará los fallos más importantes.

- En la mayoría de los casos el proceso de pruebas no encontrará todos los defectos del sistema (ver Principio 2), pero los defectos más importantes deberían ser detectados.

Esto en sí no prueba la calidad del sistema software.

- La funcionalidad del software puede no satisfacer las necesidades y expectativas de los usuarios.
- No se puede introducir la calidad a través de las pruebas, ella tiene que construirse desde el principio.

I - Fundamentos de Pruebas Software

03 - Los siete principios generales del proceso de pruebas de software.

Resumen:

- Las pruebas pueden ayudar a detectar defectos en el software, sin embargo las mismas no pueden demostrar la ausencia de defectos.
- Salvo en casos triviales, las pruebas exhaustivas son imposibles, las pruebas de muestra son necesarias.
- Las pruebas tempranas ayudan a reducir costos dado que los defectos descubiertos en fases tempranas del proceso software son corregidos con menor esfuerzo.
- Los defectos se presentan agrupados. El encontrar un defecto en una ubicación determinada significa que probablemente se encontrará otro defecto a su alrededor.
- Repetir pruebas idénticas no genera nueva Información.
- Cada entorno particular determina la forma en la cual se ejecutarán o desarrollarán las pruebas.
- Un software libre de errores no implica que sea adecuado para el uso.

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- I/01 ¿Por qué es necesario Probar?
- I/02 ¿Qué son pruebas?
- I/03 Los siete Principios generales del proceso de pruebas de software.
- **I/04 Proceso de pruebas básico.**
- I/05 Psicología en el proceso de pruebas.
- I/06 Código de Ética

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

El proceso de pruebas como proceso dentro del proceso de desarrollo software

Dependiendo del enfoque seleccionado, el proceso de pruebas tendrá lugar en diferentes puntos del proceso de desarrollo.

- Las pruebas constituyen un proceso en sí mismas.
- El proceso de pruebas está determinado por las siguientes fases:
 - Planificación de pruebas.
 - Análisis de pruebas y diseño de pruebas.
 - Implementación de pruebas y ejecución de pruebas.
 - Evaluación del criterio de finalización de pruebas y generación de informes de pruebas.
 - Actividades de cierre de pruebas.

Así también:

- Control de las pruebas (todas las fases).

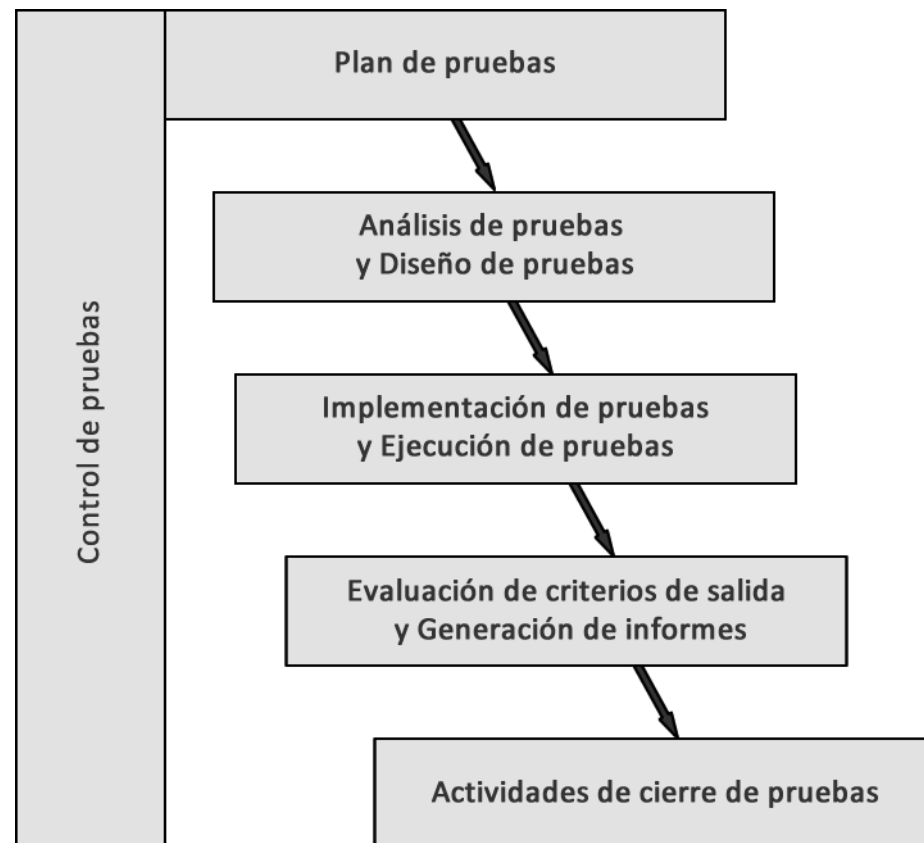
LAS FASES DEL PROCESO DE PRUEBAS SE PODRÁN SUPERPONER.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

El proceso de pruebas a lo largo del proceso de desarrollo software

- Incluye superposición y vuelta atrás (back tracking).
- ¡El proceso de pruebas es más que la ejecución de pruebas!
- Cada fase del proceso de pruebas tiene lugar de forma concurrente con las fases del proceso de desarrollo software.

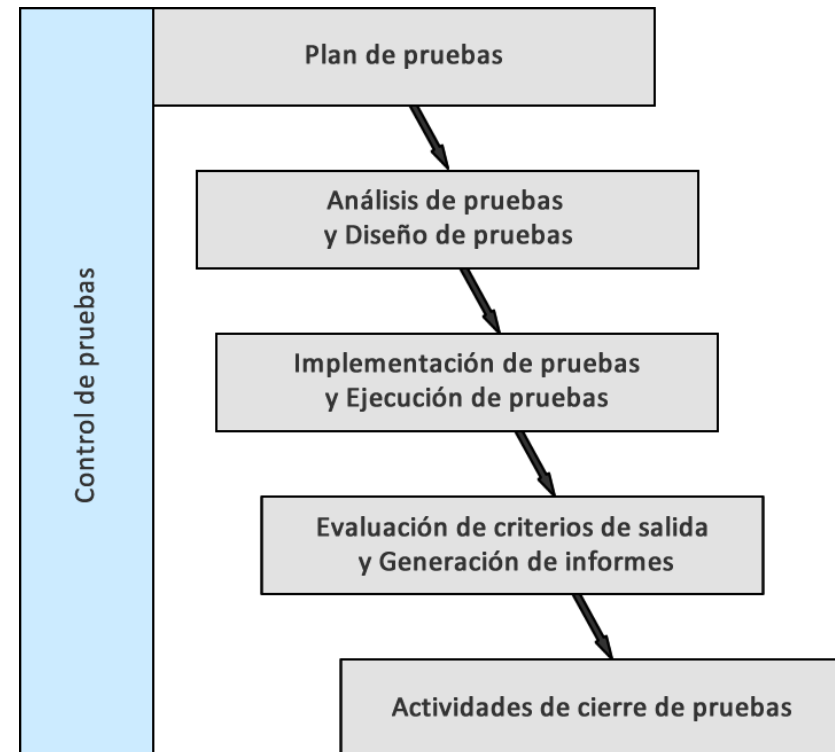


I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Control de pruebas - tareas principales

- El control de pruebas es una actividad continua que influye en la planificación de las pruebas. El plan de pruebas puede ser modificado en función de la información adquirida a partir del control de pruebas.
- El estado del proceso de pruebas se determina comparando el progreso logrado con respecto al plan de pruebas. Se iniciarán aquellas actividades que se consideraran consecuentemente necesarias.
- Se miden y analizan resultados.
- El progreso de las pruebas, la cobertura de las pruebas y el cumplimiento del criterio de finalización de pruebas son objeto de seguimiento y son documentados.
- Se inician medidas correctivas.
- Preparar y tomar decisiones.

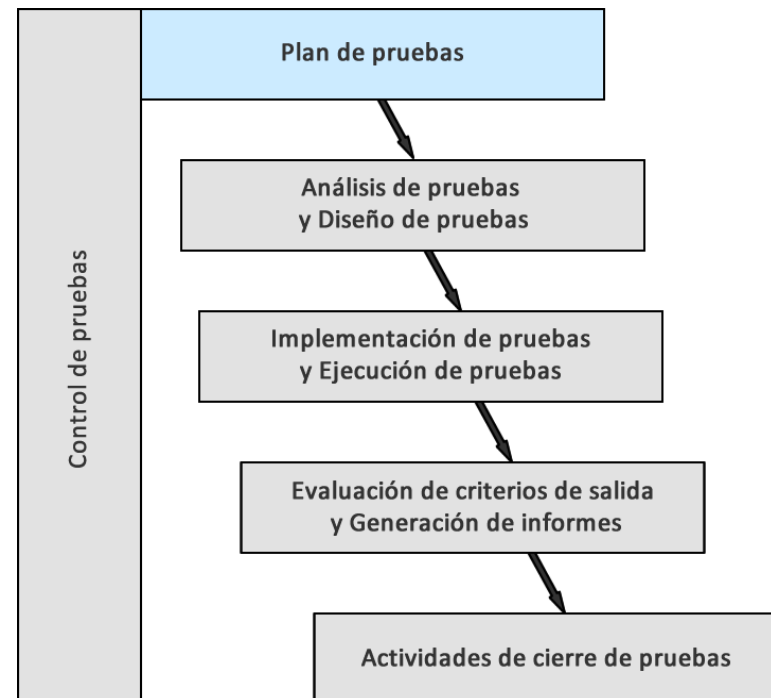


I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Planificación de pruebas - tareas principales

- Determinar el alcance y los riesgos.
- Identificar los objetivos de las pruebas y el criterio de finalización de pruebas.
- Determinar el enfoque: técnicas de pruebas, cobertura de pruebas, equipo de pruebas.
- Implementar el método de pruebas / estrategia de pruebas, planificación del tiempo disponible para las actividades a seguir.
- Adquirir/obtener y programar recursos requeridos por las pruebas: personal, entorno de pruebas, presupuesto de pruebas.



I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Términos: Planificación de pruebas

Plan Maestro de prueba (en alemán: Testkonzept):

Un documento que describe el alcance, el enfoque, los recursos y el calendario de actividades de la prueba prevista. Esto incluye, pero no se limita a los recursos, los elementos de prueba, las características de la prueba y planes de contingencia.

Estrategia de pruebas:

Una descripción de alto nivel de los niveles de prueba a realizar y las pruebas dentro de los niveles de una organización o programa (uno o más proyectos).

Enfoque de pruebas:

La aplicación de la estrategia de prueba para un proyecto específico. Por lo general incluye las decisiones que siguen a los objetivos (de prueba) del proyecto y el análisis de riesgos, puntos de partida sobre el proceso de prueba, las técnicas de diseño de la prueba que deben aplicarse, los criterios de salida y tipos de prueba para llevar a cabo.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Términos: Planificación de pruebas

Criterios de salida (después de Gilb y Graham):

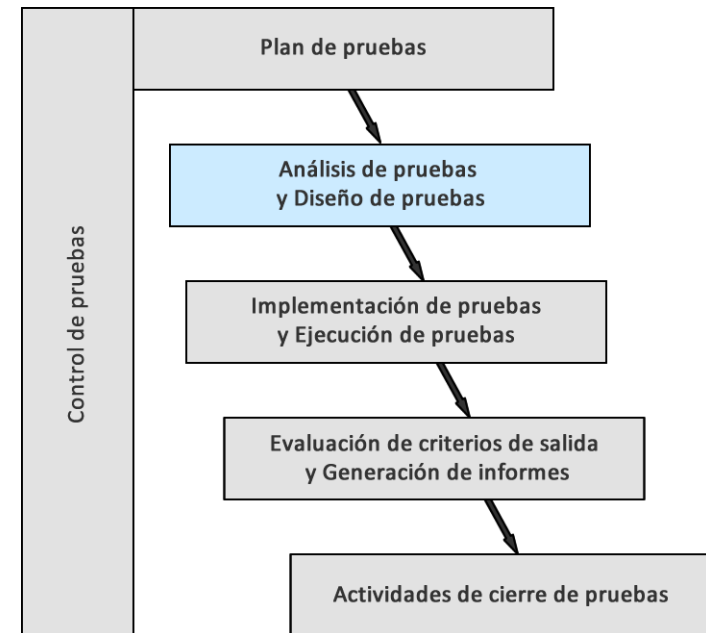
El conjunto de las condiciones genéricas y específicas, acordadas con las partes interesadas, para permitir que un proceso sea completado oficialmente. El propósito de los criterios de salida es evitar que una tarea sea considerada completa cuando todavía hay partes de la tarea pendiente que no se han terminado. Criterios de salida se utilizan para informar y para planificar cuándo dejar de probar. Esto debe hacerse para cada nivel de prueba.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Análisis y diseño de pruebas - tareas principales

- Revisión de las bases de las pruebas (requisitos, arquitectura del sistema, diseño, interfaces).
- Análisis de la arquitectura del sistema, diseño del sistema incluyendo las interfaces entre los objetos de prueba.
- Identificar condiciones de prueba específicas y los datos de prueba necesarios.
- Evaluar la disponibilidad de datos de prueba y/o la viabilidad de generar datos de prueba
- Diseño de pruebas / casos de prueba.
- Crear casos de prueba lógicos (casos de prueba con datos de prueba sin valores específicos) y establecer un orden de prioridad para los mismos.
- Los casos de prueba positivos comprueban la funcionalidad, los casos de prueba negativos comprueban situaciones en las cuales se debe realizar un tratamiento de errores
- Análisis de la posibilidad de realizar pruebas sobre el sistema.



I - Fundamentos de Pruebas Software
04 - Proceso de pruebas básico

Análisis y diseño de pruebas - tareas principales

- **Organización del entorno de pruebas (cama de pruebas - test bed).**
 - Disponibilidad (exclusiva) del entorno de pruebas, ventanas de tiempo. etc.
 - Definir el modo de operación del entorno de pruebas, incluida la administración de usuarios.
 - Carga de datos de prueba y parámetros del sistema.
 - Conexión del entorno de pruebas con los sistemas adyacentes.
- **Infraestructura de pruebas y herramientas de pruebas (si fuera necesario).**
 - Procesos, procedimientos y responsabilidades.
 - Elección, suministro. Instalación y operación de herramientas de pruebas.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Términos: Análisis y diseño de pruebas

Datos de prueba (test data):

Datos que existen en el sistema antes de que una prueba sea ejecutada y que afecta o es afectado por el componente o sistema sujeto a pruebas.

Datos de entrada (input data):

Variable que es leída por un componente (tanto almacenada dentro del sistema como fuera del mismo).

Cobertura de pruebas (test coverage):

Grado en el cual un elemento específico ha sido ejecutado por un conjunto de pruebas. Utilizado con mayor frecuencia en pruebas de caja blanca con el objeto de determinar la cobertura del código.

I - Fundamentos de Pruebas Software
04 - Proceso de pruebas básico

Términos: Análisis y diseño de pruebas

Oráculo de pruebas (test oracle):

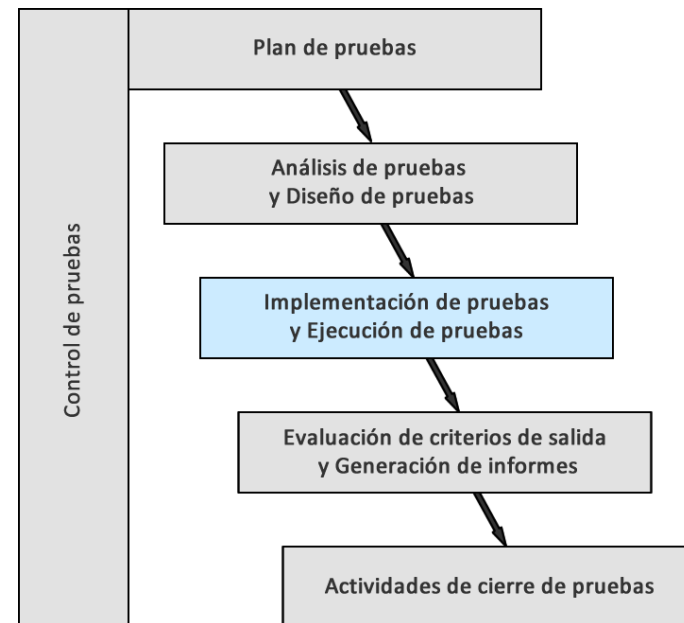
Fuente que permite determinar los resultados esperados de un software sujeto o pruebas: comparativas (benchmarks) (también resultado de pruebas previas), manuales de usuario o conocimiento especializado. No debe ser el código.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Ejecución de pruebas - tareas principales

- Desarrollo y asignación de un orden de prioridad a los casos de prueba.
 - Creación de datos de prueba, desarrollo de procedimientos de prueba.
 - Creación de secuencias de pruebas (juegos de pruebas - test suites)
- Creación de scripts de automatización de pruebas, si fuera necesario.
- Configuración del entorno de pruebas (cama de pruebas - test bed).
- Ejecución de pruebas (de forma manual o automática)
 - Seguimiento de secuencias de pruebas establecidas en el plan de pruebas (juegos de pruebas orden de los casos de prueba).



I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Ejecución de pruebas - tareas principales

- Registro y análisis de los resultados de pruebas.
- Reiteración de pruebas (retest)
 - Después de corrección de defecto
- Pruebas de regresión (regression testing).
 - Tras una corrección o instalar una nueva funcionalidad (cambio): Repetición de pruebas con el objeto de asegurar que los defectos han sido corregidos y que las correcciones o modificaciones no han Introducido nuevos defectos.

I - Fundamentos de Pruebas Software
04 - Proceso de pruebas básico

Términos: Ejecución de pruebas

Juego de pruebas (test suite) / secuencia de pruebas (test sequence):

Conjunto de casos de prueba para un componente o sistema, donde la poscondición de un caso de prueba es utilizada como precondition para el caso de prueba siguiente.

Especificación de procedimiento de pruebas (test procedure specification, escenario de pruebas - test scenario):

Documento en el cual se especifica una secuencia de acciones para la ejecución de una prueba. También conocido como guión de pruebas o guión de pruebas manuales (Según IEEE Std 829)

Ejecución de pruebas (test execution):

Es el proceso de llevar a cabo una prueba, aportando / generando los resultados reales.

Registro de pruebas (test log) (informe de pruebas - test report):

Relación cronológica de los detalles relevantes relativa a la ejecución de pruebas cuando se desarrollaron las pruebas, qué resultados fueron generados.

I - Fundamentos de Pruebas Software
04 - Proceso de pruebas básico

Términos: Ejecución de pruebas

Pruebas de regresión (regression tests):

Pruebas tras la modificación de un programa previamente probado, con el objeto de asegurar que no se han introducido o descubierto defectos en áreas que no hubieran sido objeto de modificación como resultado de los cambios realizados. Se realizan cuando el software o su entorno han sido modificados.

Pruebas de confirmación (confirmation testing), reiteración de pruebas (retest):

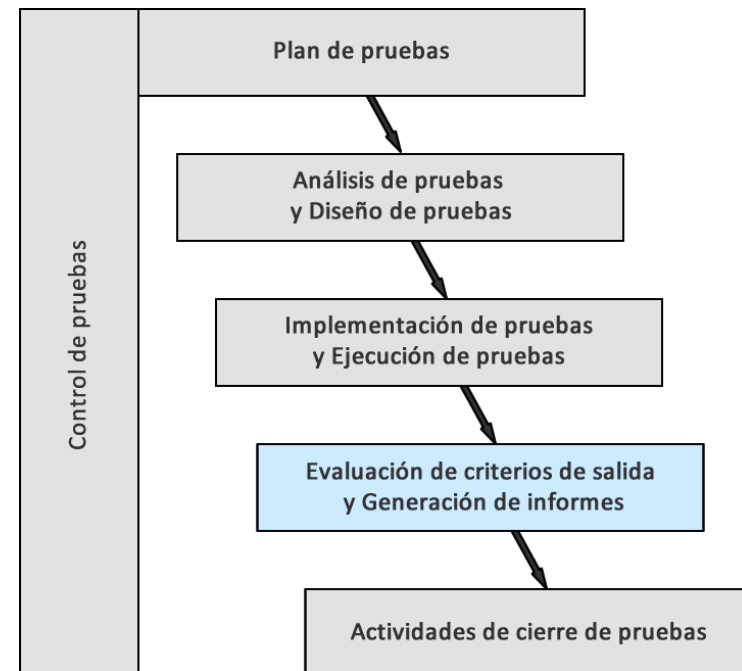
Repetición de una prueba después de la corrección de un defecto, con el objeto de confirmar que el defecto ha sido eliminado con éxito.

I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Evaluación del criterio de salida - tareas principales

- Evaluación de la ejecución de pruebas con respecto a objetivos definidos.
- Evaluación de los registros de pruebas (resumen de las actividades de pruebas, resultados de prueba, comunicar criterio de salida).
- Proporcionar información con el objeto de permitir la decisión con respecto a si tendrán lugar pruebas adicionales.

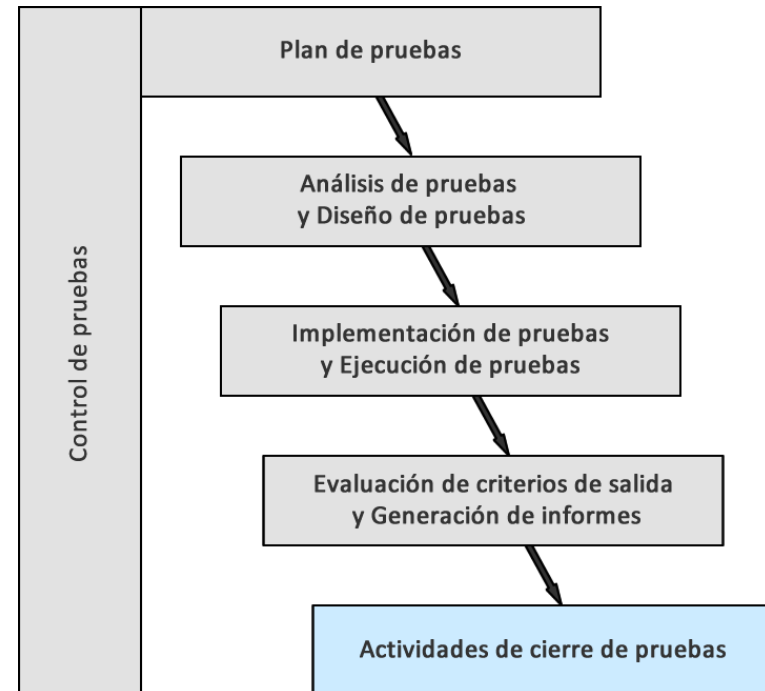


I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Actividades de cierre de pruebas - tareas principales

- Recopilar datos de las actividades del proceso de pruebas finalizadas con el objeto de consolidar la experiencia, utensilios de pruebas (testware), hechos y resultados (valores).
- Cierre de informes de incidencias o generación de solicitudes de cambio para cualquier punto que permaneciera abierto.
- Comprobar qué entregables planificados han sido entregados y probados.
- Documentar la aceptación del sistema.
- Finalizar y archivar los utensilios de pruebas (testware), el entorno de pruebas y la infraestructura de pruebas para un uso posterior, transferencia al entorno de operaciones.
- Analizar las lecciones aprendidas para futuros proyectos.



I - Fundamentos de Pruebas Software

04 - Proceso de pruebas básico

Resumen:

El proceso de pruebas se puede dividir en diferentes fases.

- **Planificación de pruebas** (test planning) abarca actividades como la definición de la estrategia de pruebas para todas las fases así como la planificación de los recursos (tiempo, personal, máquinas).
- **Diseño de pruebas** (especificación) abarca el diseño de casos de prueba y sus resultados esperados.
- **Ejecución de pruebas** abarca la definición de datos de pruebas, la ejecución de pruebas y la comparación de resultados.
- **Informes de pruebas** (logging) registro de los resultados de pruebas, en forma tabulada, en base de datos o de forma escrita.
- **Evaluación de pruebas** abarca el análisis de defectos y la evaluación del criterio de salida.
- **Control de pruebas** consiste en las actividades de control que abarcan todas las fases anteriores.

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- I/01 ¿Por qué es necesario Probar?
- I/02 ¿Qué son pruebas?
- I/03 Los siete Principios generales del proceso de pruebas de software.
- I/04 Proceso de pruebas básico.
- **I/05 Psicología en el proceso de pruebas.**
- I/06 Código de Ética

I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Roles y responsabilidades

Rol: Desarrollador

- Implementa requisitos.
- Desarrolla estructuras.
- Desarrolla el software.
- Crea un producto.

Rol: Probador (Tester)

- Planifica las actividades de pruebas.
- Diseña casos de prueba.
- Su única preocupación es encontrar defectos.
- Encontrar errores producidos por un desarrollador es su éxito.

Percepción:

La actividad del desarrollador es
CONSTRUCTIVA

La actividad del probador (tester) es
DESTRUCTIVA

¡Las pruebas también constituyen una actividad constructiva, su propósito es la eliminación de defectos en un producto!

I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Características personales de un buen tester /1

- Curioso, perceptivo, atento a los detalles.
 - Con el objeto de comprender los escenarios prácticos del cliente.
 - Con el objeto de poder analizar la estructura de una prueba.
 - Con el objeto de descubrir dónde se pueden manifestar fallos.

- Escéptico y con actitud crítica.
 - Los objetos de prueba contienen defectos. Usted solo debe encontrarlos.
 - No se debe tener temor al hecho de que pueden ser descubiertos defectos serios que tengan un impacto sobre la evolución del proyecto.

I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Características personales de un buen tester /2

- **Aptitudes para la comunicación.**
 - Necesarias para llevar malas noticias a los desarrolladores.
 - Necesarias para vencer estados de frustración.
 - Tanto cuestiones técnicas como prácticas, relativas al uso del sistema, deben ser entendidas y comunicadas.
 - Una comunicación positiva puede ayudar a evitar o facilitar situaciones difíciles.
 - Facilitan establecer una relación de trabajo con los desarrolladores a corto plazo.
- **Experiencia.**
 - Factores personales influyen en la ocurrencia de errores.
 - La experiencia ayuda a identificar dónde se pueden acumular errores.

I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Diferencias: diseñar - desarrollar - probar

- Las pruebas requieren una perspectiva distinta a la del diseño y desarrollo de sistemas software.
 - **Objetivo común:** proveer un buen producto software.
 - **Cometido del diseño:** ayudar al cliente a proveer/suministrar los requisitos correctos.
 - **Cometido de los desarrolladores:** convertirlos requisitos en funciones.
 - **Cometido de los probadores (testers):** evaluar la correcta implementación de los requisitos del cliente.
- En principio, una persona puede asumir los tres roles en su trabajo.
 - Se deben tener en cuenta las diferencias en objetivos y modelos de conducta.
 - Es difícil pero posible.
 - A veces otras soluciones (pruebas independientes) pueden ser más sencillas y aportar mejores resultados.

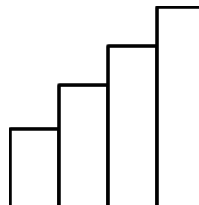
I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Pruebas independientes

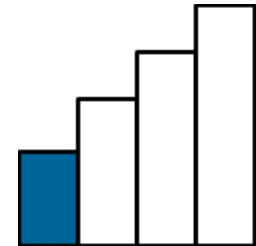
La separación de las responsabilidades en el proceso de pruebas apoya/promueve la evaluación independiente de los resultados de las pruebas.

En el siguiente diagrama se representa el grado de independencia a través de un gráfico de barras.



I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Tipos de organización de pruebas /1

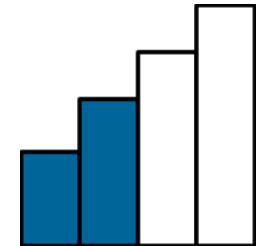


- **Pruebas del desarrollador**

- El desarrollador nunca analizará su "creación" de forma imparcial (apego afectivo).
 - Sin embargo, él conoce el objeto de pruebas mejor que nadie.
 - Habrá costos adicionales debido a la formación/Información de otras personas respecto del objeto de pruebas.
- Las personas tienden a pasar por alto sus propios defectos.
 - Los desarrolladores corren el riesgo de no reconocer defectos evidentes.
- Errores cometidos como consecuencia de una mala interpretación de los requisitos se mantendrán sin ser detectados.
 - Establecer grupos de prueba donde los desarrolladores prueben los productos de otros, ayuda a evitar, o al menos, reducir la posibilidad de ocurrencia de esto tipo de anomalía (producida por negligencia).

I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Tipos de organización de pruebas /2

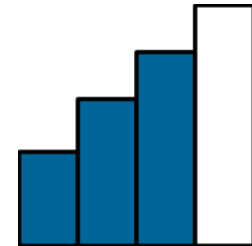


- **Equipos de desarrollo**

- Los desarrolladores hablan el mismo lenguaje.
- Los costos de formación/información en lo relativo a objetos de prueba se mantienen a un nivel moderado, especialmente cuando los equipos intercambian objetos de prueba.
- Peligro de generación de conflictos entre equipos de desarrollo.
 - Un desarrollador que busca y encuentra un defecto no será el mejor amigo del autor del objeto de prueba analizado.
- Mezcla de actividades de desarrollo y pruebas.
 - Cambios frecuentes en la forma de pensar.
 - Dificultad en el control del presupuesto del proyecto.

I - Fundamentos de Pruebas Software
05 - Psicología en el proceso de pruebas

Tipos de organización de pruebas /3



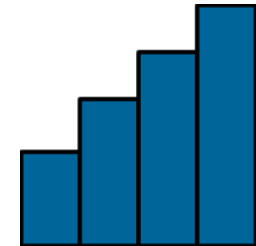
- **Equipos de pruebas**

- La creación de equipos de pruebas que den servicio o diferentes áreas de proyecto mejora la calidad de las pruebas.
- Es importante que los equipos de pruebas de diferentes áreas en el proyecto trabajen de forma independiente.

I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Tipos de organización de pruebas /4



- **Outsourcing de pruebas (subcontratación)**

- La separación de las actividades de pruebas y desarrollo aportan la máxima independencia entre los objetos de prueba y el probador (tester).
- Las actividades de pruebas subcontratadas (externalizadas) son ejecutadas por personal con un conocimiento relativamente pequeño de los objetos de prueba y de los antecedentes del proyecto.
 - Lo curva de aprendizaje Implica altos costos, por lo tanto deberían ser Incorporados expertos Independientes en etapas tempranas del proyecto.
- Los expertos externos cuentan con un alto nivel de conocimiento (know how) del proceso de pruebas.
 - Está asegurando un diseño de pruebas apropiado.
 - Se alcanza la optimización en el uso de métodos y herramientas.

- **Diseño de casos de prueba de forma automática**

- Generación de casos de prueba asistida por ordenador, por ejemplo casos de prueba basados en documentos de especificaciones formales.

I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Dificultades /1

- Incapacidad de comprensión mutua.
 - Los desarrolladores deberían contar con un conocimiento básico de pruebas.
 - Los probadores (testers) deberían contar con un conocimiento básico de desarrollo software.
- Especialmente en situaciones de tensión, la detección de errores cometidos por alguien frecuentemente conduce a conflictos.
 - La forma de documentar los defectos y la forma en la cual el defecto es descrito determinará cómo se desarrollarán los hechos.
 - Las personas no deberían ser criticadas, los defectos deben ser descritos en términos objetivos.
 - La descripción de los defectos debería ayudar al desarrollador a encontrar el error.
 - Los objetivos comunes siempre deben ser la cuestión principal.

I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Dificultades /2

- La comunicación entre probadores (testers) y desarrolladores es insuficiente o inexistente. Este hecho puede hacer imposible el trabajo conjunto.
 - Los probadores (testers) son vistos únicamente como "portadores de malas noticias".
 - Mejora: intente ponerse en el lugar (rol) de la otra persona. ¿Ha llegado mi mensaje? ¿La respuesta es suficiente?
- Un proceso de pruebas sólido requiere la distancia apropiada con respecto al objeto de prueba.
 - Se adquiere un punto de vista independiente e imparcial a través de la distancia con respecto al desarrollo.
 - Sin embargo, una distancia muy grande con respecto al objeto de prueba y el equipo de desarrollo conducirá a mayores esfuerzos y tiempo para las pruebas.

I - Fundamentos de Pruebas Software

05 - Psicología en el proceso de pruebas

Resumen

- Las personas cometen errores, toda implementación tiene defectos.
- La naturaleza humana dificulta la posibilidad de hacer frente a los defectos propios (ceguera a los errores).
- Desarrollador y probador (tester) implican el encuentro de dos mundos distintos.
 - El desarrollo es constructivo - algo que no estaba ahí previamente es creado.
 - El proceso de pruebas resulta destructivo a primera vista - ¡Se delectarán defectos!
 - Juntos, el desarrollo y las pruebas son constructivos en su objetivo de obtener un producto de software con la menor cantidad de defectos posible.
- Las pruebas Independientes aumentan la calidad del proceso de pruebas.
 - En lugar de equipos de desarrolladores utilice equipos de prueba o equipos de prueba con personal externo.

I - Fundamentos de Pruebas Software

Agenda

Capítulo I - Fundamentos de Pruebas Software

- I/01 ¿Por qué es necesario Probar?
- I/02 ¿Qué son pruebas?
- I/03 Los siete Principios generales del proceso de pruebas de software.
- I/04 Proceso de pruebas básico.
- I/05 Psicología en el proceso de pruebas.
- **I/06 Código de Ética**

I - Fundamentos de Pruebas Software

06 – Código de Ética

Código de Ética / 1

Las personas que participan en pruebas de software, tienen acceso a información muy privilegiada y crítica. Para garantizar que la información no es objeto de un uso inadecuado, el código de ética es necesario.

Público: Los probadores certificados de software actuarán según los intereses de su cliente y su empleador, de conformidad con el interés público, especialmente al trabajar en un sistema de seguridad crítico

Cliente y Empleador: El probador certificado deberá actuar según los intereses de sus clientes y empleadores, de acuerdo con el interés público. Por ejemplo, que no se filtre información interna o privada, sobre el cliente o el empleador en Internet.

I - Fundamentos de Pruebas Software

06 – Código de Ética

Código de Ética / 2

Producto: Los probadores certificados de software asegurarán que las prestaciones que ofrecen (a los productos y sistemas que ponen a prueba) cumplen con los estándares profesionales más altos posibles. Es decir, el trabajo como consultor de no dejar los detalles importantes para el cliente.

Sentencia: Los probadores certificados de software deberán mantener la integridad e independencia en su juicio profesional. Tal vez un jefe de proyecto está pidiendo ocultar defectos importantes al cliente, lo que constituye una falta a la independencia y a la ética.

Gestión: Los gerentes y líderes certificados en pruebas de software promoverán un enfoque ético a la gestión de pruebas de software. Favorecer a un probador sobre otro para obtener beneficios personales es una seria falta a la ética empresarial.

I - Fundamentos de Pruebas Software

06 – Código de Ética

Código de Ética / 3

Profesión: Los probadores certificados de software deberán promover la integridad y reputación de la profesión de acuerdo con el interés público.

Colegas: Los probadores certificados de software serán justos y apoyarán a sus colegas y promoverán la cooperación con los desarrolladores de software.

Personal: Los probadores de software certificado se capacitarán permanentemente sobre la práctica de su profesión y promoverán un enfoque ético de la práctica de la profesión. Asistir a cursos y leer libros puede ser una manera de mantener el conocimiento en un alto nivel.

II - Pruebas a lo largo del ciclo de vida software

Agenda

Capítulo II - Pruebas a lo largo del ciclo de vida software

- **II/01 Modelos de desarrollo software.**
- II/02 Niveles de prueba.
- II/03 Tipos de prueba - objetivos de las pruebas.
- II/04 Pruebas de Mantenimiento

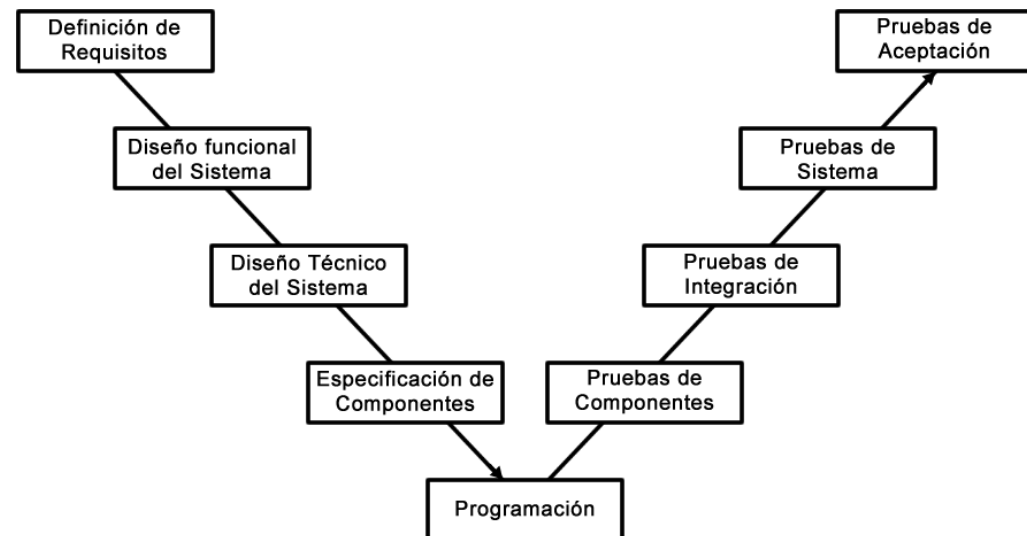
II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Pruebas a lo largo del modelo-V general (Modelo secuencial de desarrollo)

- El modelo-V general es el modelo de desarrollo software más utilizado.
- Desarrollo y pruebas tienen dos ramas iguales.
 - Cada nivel de desarrollo tiene su correspondiente nivel de pruebas.

- Las pruebas (rama derecha) se diseñan en paralelo al desarrollo software (rama izquierda).
- Las actividades del proceso de pruebas tienen lugar a lo largo de todo el ciclo de vida software.



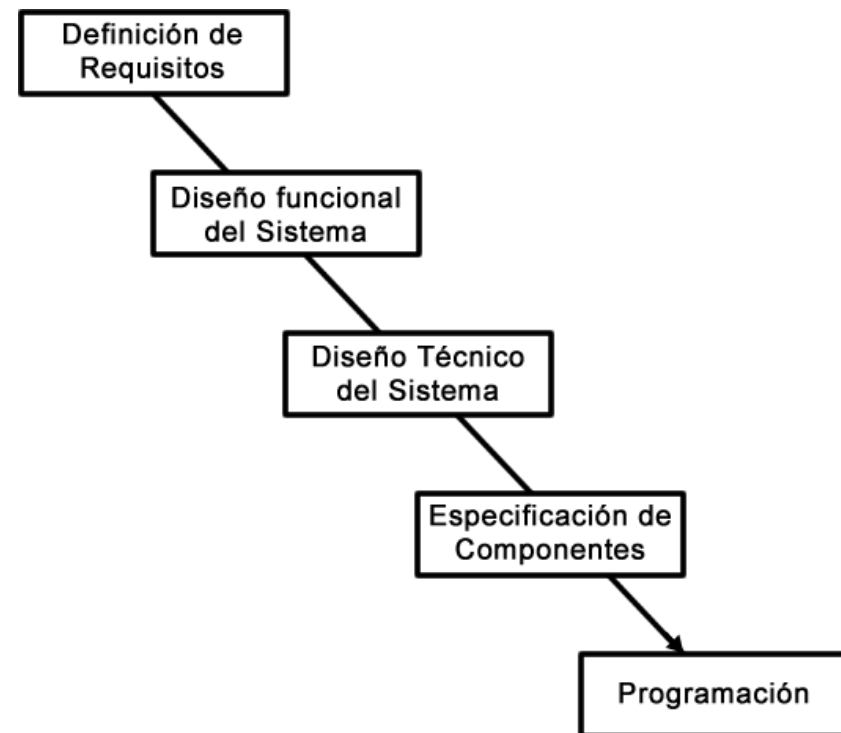
II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Pruebas a lo largo del modelo-V general

Rama: desarrollo software.

- Definición de requisitos.
 - Documentos de especificación.
- 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.



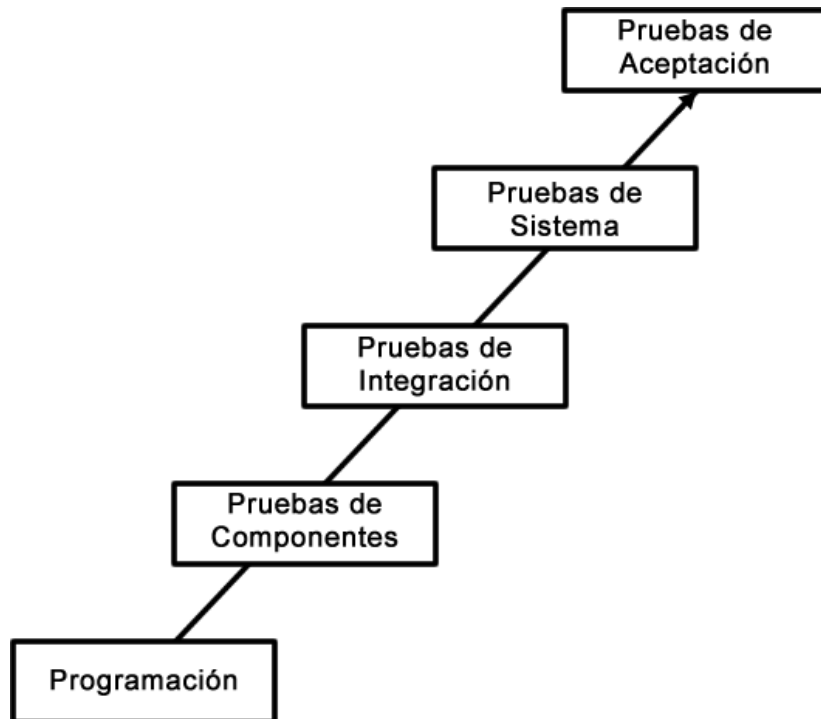
II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Pruebas a lo largo del modelo-V general

Rama: pruebas software.

- Pruebas de aceptación.
 - Pruebas formales de los requisitos del cliente.
- Pruebas de sistema.
 - Sistema Integrado, especificaciones.
- Pruebas de integración.
 - Interfaces de componentes.
- Pruebas de componente.
 - Funcionalidad del componente.



II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Verificación vs. Validación

- Verificación
 - Comprobación de la conformidad con los requisitos establecidos (definición según ISO 9000).
 - Cuestión clave: ¿Se ha procedido correctamente en la construcción del sistema?
 - ¿Hemos sumado 1 más 1 correctamente?

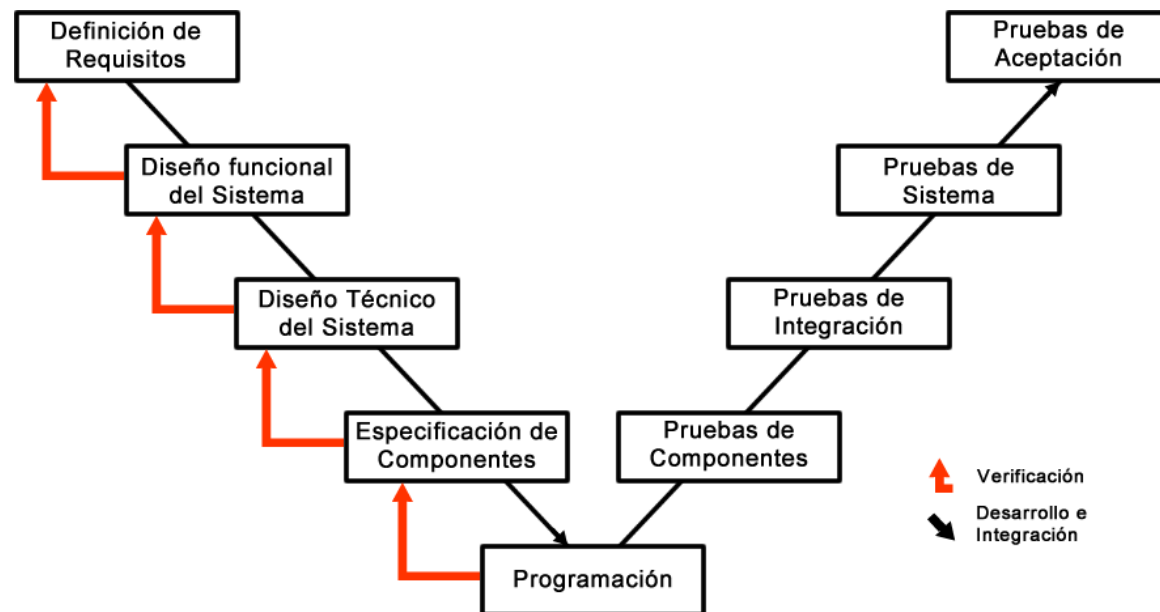
- Validación
 - Comprobación de la idoneidad para el uso esperado (definición según ISO 9000).
 - Cuestión clave: ¿Hemos construido el sistema de software correcto?
 - ¿El objetivo era sumar 1 más 1 o deberíamos haber restado?

II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Verificación dentro del modelo-V general

- Cada nivel de desarrollo se verifica respecto de los contenidos del nivel que le precede.
 - Verificar: comprobar la evidencia.
- Verificar significa comprobar si los requisitos y definiciones de niveles previos han sido Implementados correctamente.



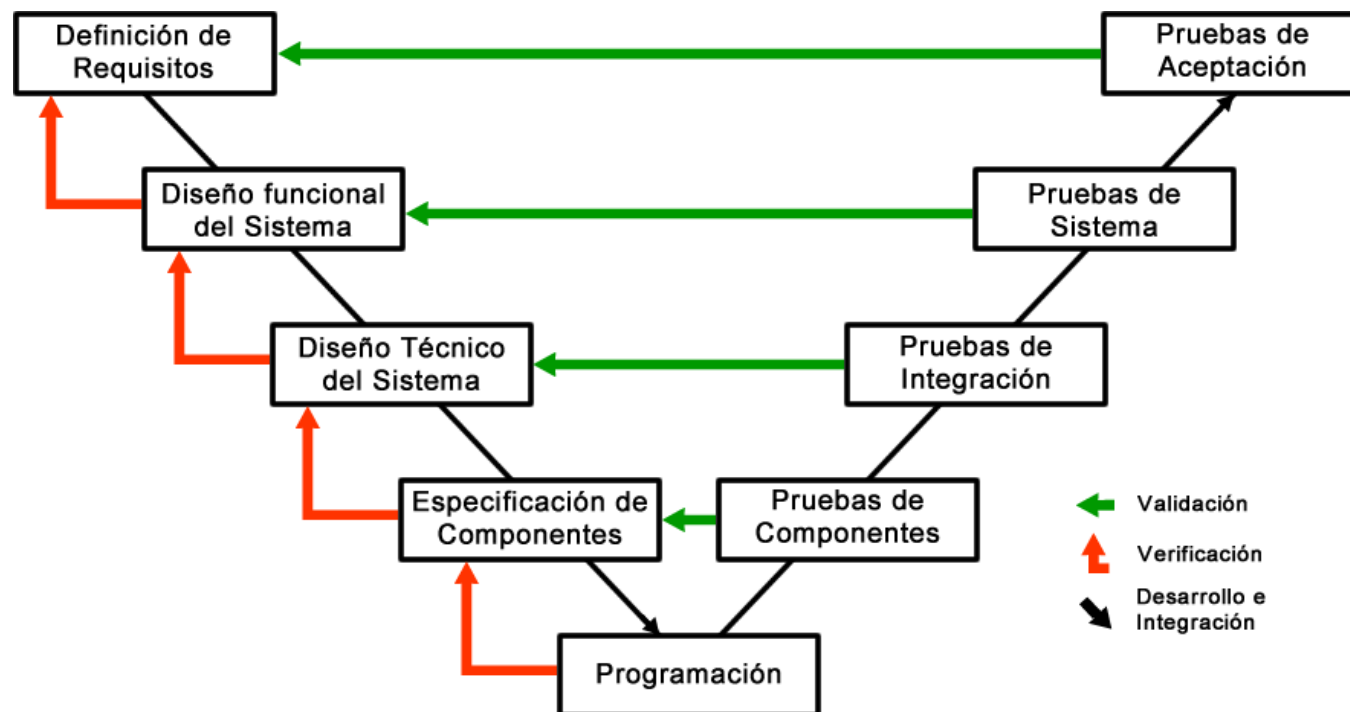
II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Validación dentro del modelo-V general

Validación dentro del modelo-V general

- Lo validación se refiere a la corrección de cada nivel de desarrollo.
 - Validar: Dar prueba de la aportación de valor.
- Validar significa comprobar lo adecuado de los resultados de un nivel de desarrollo.

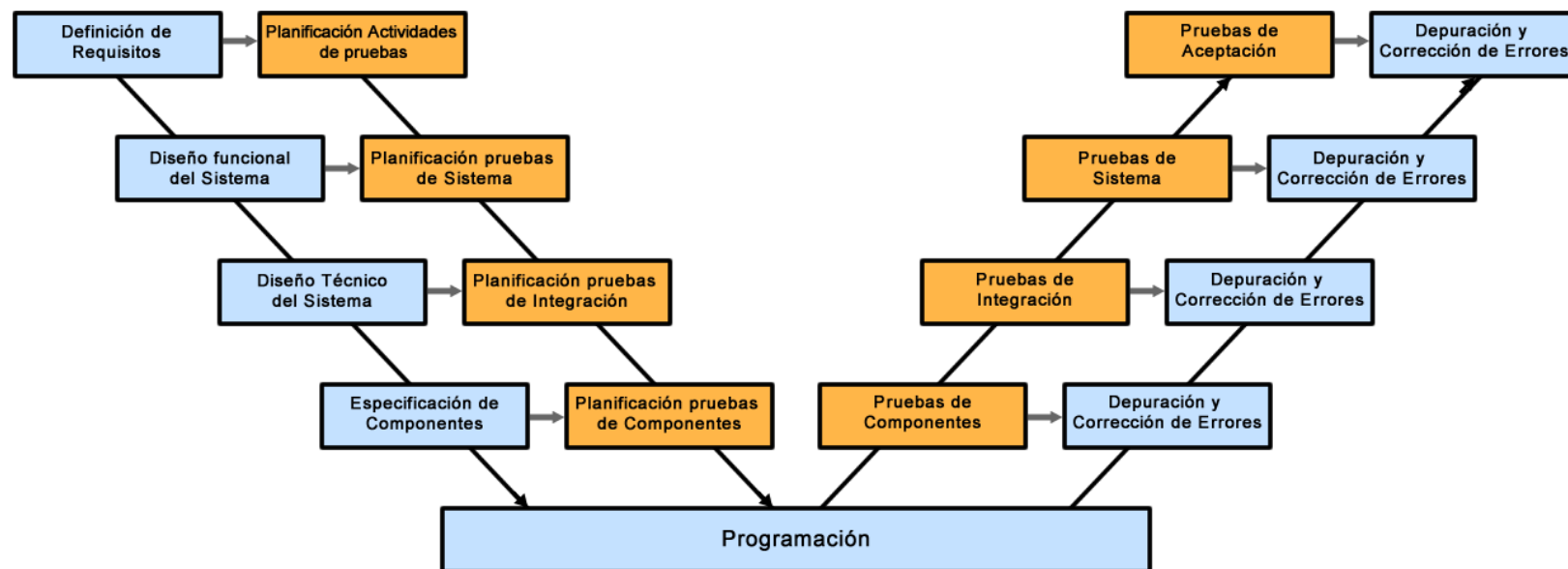


II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Modelo-W

- El modelo W puede ser visto como una extensión del modelo-V general
- En los estados W-modelo, ciertas actividades de control de calidad se realizará en paralelo con el proceso de desarrollo



II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Otros modelos de prueba: modelos iterativos /1

- **Desarrollo software iterativo.**

- Las actividades: Definición de requisitos, diseño, desarrollo y pruebas se segmentan en pasos reducidos y se ejecutan de forma continua.
- Se debe lograr el consentimiento con el cliente tras cada iteración con el objeto de modificar el rumbo del proyecto si fuera necesario.

- **Ejemplos de modelos iterativos:**

- **Modelo Prototipado:** Desarrollo rápido de una representación del sistema que pudiera ser objeto de uso. Seguido de modificaciones sucesivas hasta que el sistema sea finalizado.
- **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.
- **Proceso Unificado (Rational Unified Process - RUP):** Modelo orientado a objetos y producto de la compañía Rational/IBM. Principalmente aporta el lenguaje de modelado UML y soporte al Proceso Unificado.
- **Programación Extrema (Extreme Programming - XP):** El desarrollo y las pruebas tienen lugar sin una especificación de requisitos formalizada.

II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Otros modelos de prueba: modelos iterativos /2

- **Características de los modelos iterativos:**
 - Cada iteración contribuye con una característica adicional del sistema bajo desarrollo.
 - Cada iteración puede ser probada por separado.
 - Las pruebas de regresión son de gran relevancia.
 - En cada iteración, la verificación (relación con el nivel precedente) y la validación (grado de corrección del producto dentro del nivel actual) se pueden efectuar por separado.

II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Otros modelos de prueba: modelos iterativos /3

- **Desarrollo guiado por pruebas (Test Driven Development - TDD)**
 - Desarrollo basado en conjunto de pruebas (Suite cases)
 - Preparar los ciclos de pruebas
 - Prueba automatizada utilizando herramientas de prueba
 - Desarrollo, según los casos de prueba
 - Preparar versiones tempranas del componente para probarlo
 - Ejecución automática de pruebas
 - Corregir los defectos en las versiones
 - Repetir el conjunto de pruebas hasta que no se encuentran errores

Primero se diseñan las pruebas, luego el software es desarrollado

II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Principios de todos los modelos

- Cada actividad de desarrollo debe ser probada.
 - Ninguna porción del software puede quedar sin ser probada, tanto si ha sido desarrollada "en una única fase" o de forma iterativa.
- Cada nivel de pruebas debería ser probado de forma específica.
 - Cada nivel de pruebas cuenta con sus propios objetivos de prueba.
 - Las pruebas llevadas a cabo en cada nivel deben reflejar estos objetivos.
- El proceso de pruebas comienza con mucha antelación a la ejecución de pruebas.
 - Tan pronto como el desarrollo comienza puede comenzar la preparación de las pruebas correspondientes.
 - También es el caso de las revisiones de documentos comenzando por los conceptos, especificación y el diseño en conjunto.

II - Pruebas a lo largo del ciclo de vida software

01 – Modelos de desarrollo software.

Resumen

- El desarrollo de software utiliza modelos tanto para el propio desarrollo de software como para las actividades del proceso de pruebas.
- El modelo más conocido es el modelo-V, el cual describe los niveles de desarrollo y niveles de prueba como dos ramas relacionadas.
- Los modelos iterativos más relevantes son RUP y XP.
- Las actividades de pruebas están recomendadas en todos los niveles de desarrollo.

II - Pruebas a lo largo del ciclo de vida software

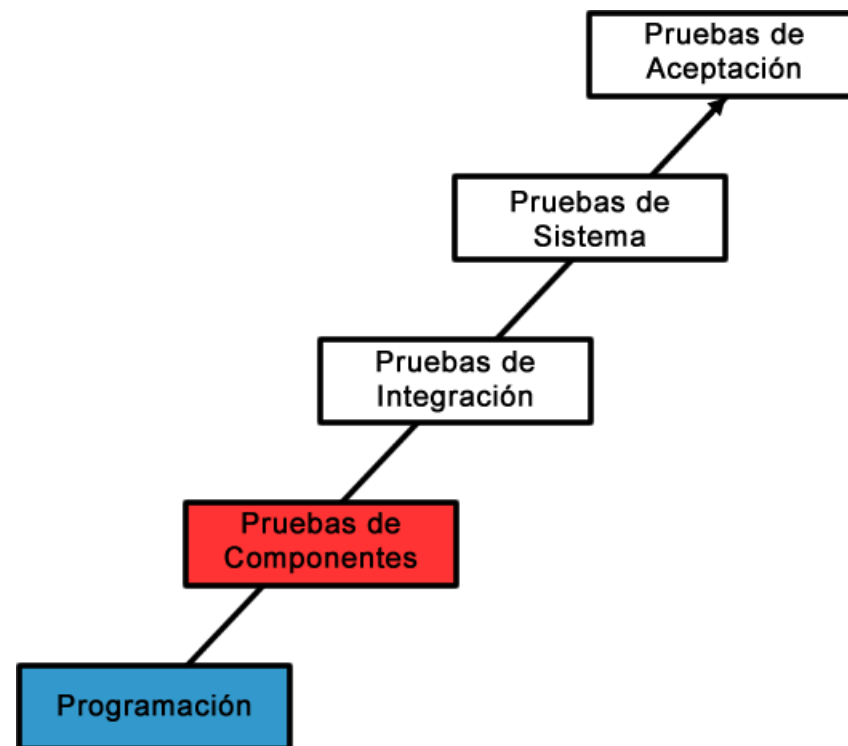
Agenda

Capítulo II - Pruebas a lo largo del ciclo de vida software

- II/01 Modelos de desarrollo software.
- **II/02 Niveles de prueba.**
- II/03 Tipos de prueba - objetivos de las pruebas.
- II/04 Pruebas de Mantenimiento

II - Pruebas a lo largo del ciclo de vida software
02 – Niveles de prueba.

Pruebas de Componente

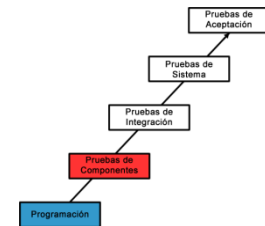


II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de componente

- Bases de prueba (Test basis):
 - Requisitos (requerimientos)de los componentes
 - Diseño detallado
 - Código
- Objetos de prueba típicos:
 - Componentes / clases / unidades / módulos
 - Programas
 - Conversión de datos / programas de migración
 - Base de datos de los módulos

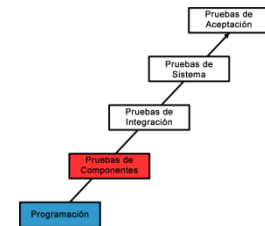


II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Componente – Definición

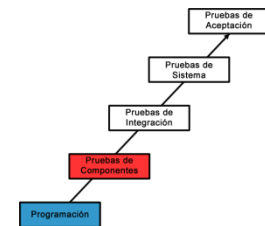
- Pruebas de componente (pruebas unitarias, pruebas de módulo, pruebas de clase, pruebas de desarrollador).
 - Pruebas de cada componente tras su realización.
- Dadas las convenciones de cada lenguaje de programación para la asignación de nombres a sus respectivos componentes, se podrá hacer referencia a un componente como:
 - Prueba de módulo (module test) (por ejemplo en C).
 - Prueba de clase (por ejemplo en Java o C++).
 - Prueba de unidad (por ejemplo en Pascal).
- Los componentes son referidos como módulos, clases o unidades. Dado que los desarrolladores posiblemente pueden estar involucrados en la ejecución de pruebas, éstas también son denominadas pruebas de desarrollador (developer test).



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de componente – Alcance

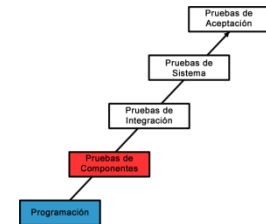


- Sólo se prueban componentes individuales.
 - Un componente puede estar constituido por un conjunto de unidades más pequeñas.
 - Los objetos de prueba no siempre pueden ser probados en solitario (de forma autónoma).
- Cada componente es probado de forma independiente.
 - Descubriendo defectos internos.
 - Los efectos cruzados entre componentes quedan fuera del alcance de estas pruebas.
- Los casos de pruebas pueden ser derivados de:
 - Especificación de componentes
 - Diseño de Software
 - Modelos de datos

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de componente - Pruebas Funcionales y No Funcionales /1.



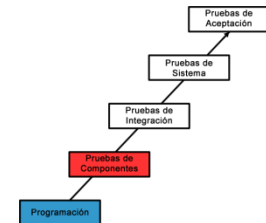
- **Pruebas de funcionalidad**

- Toda funcionalidad debe ser probada, por lo menos, con un caso de prueba.
 - Las funciones: ¿Se realizan correctamente? La funcionalidad: ¿Cumple todas las especificaciones?
- Defectos descubiertos habitualmente:
 - Defectos en el tratamiento de datos, normalmente en los valores fronteras (boundary values).
 - Funciones ausentes (missing functions).

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

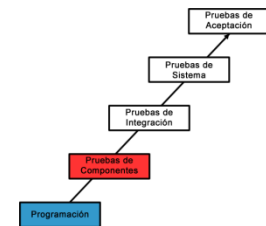
Pruebas de componente - Pruebas Funcionales y No Funcionales /2.



- **Probar la robustez (testing robustness) (resistencia a datos de entrada inválidos).**
 - Los casos de prueba que comprueban datos de entrada Inválidos se denominan pruebas negativas
 - Un sistema robusto aporta un tratamiento apropiado para datos de entrada erróneos.
 - La aceptación por parte del sistema de datos de entrada erróneos puede producir fallos en un futuro procesamiento de los mismos (datos de salida erróneos, fallo del sistema - system crash).
- **Otros atributos No Funcionales pueden ser probados**
 - Por ejemplo pruebas de estrés y rendimiento

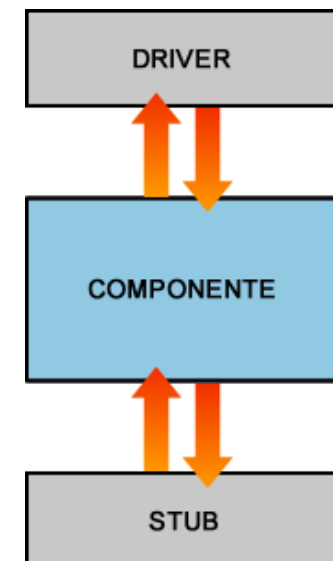
II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.



Pruebas de componente - Arnés de pruebas

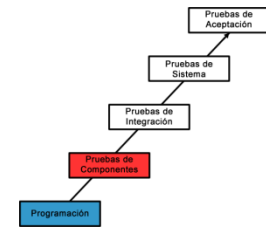
- La ejecución de pruebas de componente requiere, frecuentemente, de **DRIVERS** y **STUBS**.
 - Un **DRIVER** procesa la interfaz de un componente.
 - Los **DRIVERS** simulan datos de entrada, registran datos de salida y aportan un arnés de pruebas (test harness).
 - Los **DRIVERS** utilizan herramientas de programación.
- Un **STUB** reemplaza o simula un componente que aún no se encuentre disponible.
- Para programar un **DRIVER** o un **STUB**, el probador debe>
 - Tener conocimientos de programación
 - Tener el código fuente disponible
 - Puede necesitar herramientas especiales



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de componente - métodos

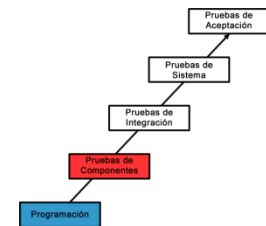


- El código fuente se encuentra disponible para el probador (tester).
 - Caso probador (tester) = desarrollador: Las pruebas se desarrollan con una fuerte orientación hacia el desarrollo.
 - El conocimiento de la funcionalidad, estructura de componentes y variables puede ser aplicado para el diseño de casos de prueba.
 - Las pruebas funcionales pueden ser aplicadas (con frecuencia).
 - Adicionalmente, el uso de depuradores (debuggers) y otras herramientas de desarrollo permitirán acceso directo a las variables del programa.
- El conocimiento del código fuente permitirá la aplicación de métodos de caja blanca (white box) para pruebas de componente.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

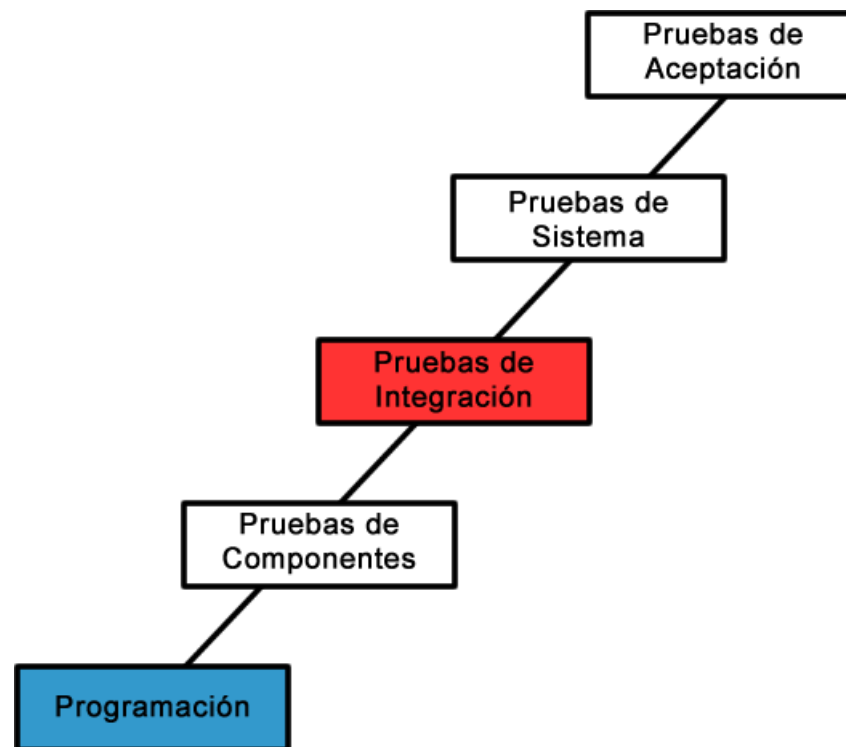
Pruebas de componente - Resumen



- 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 reemplazadas por **STUBS**.
- Las pruebas de componente podrán comprobar características funcionales y no funcionales de un sistema.

II - Pruebas a lo largo del ciclo de vida software
02 – Niveles de prueba.

Pruebas de Integración

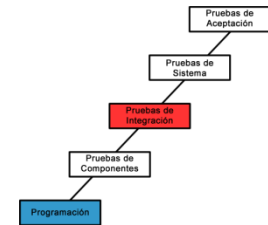


II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración

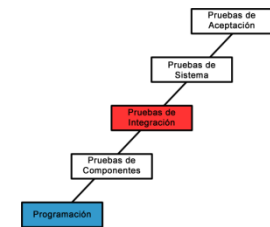
- Base de la prueba:
 - Software y diseño de sistemas
 - Arquitectura
 - Diagramas de Flujo
 - Casos de Uso
- Objetos de prueba típicos:
 - Sub-sistemas de implementación de bases de datos
 - Infraestructura
 - Interfaces
- Configuración del sistema:
 - Configuración de datos



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Integración – Definición

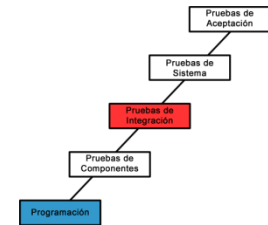


- También llamadas pruebas de interfaz
- Comprueba la interacción entre elementos software (componentes) tras la integración del sistema.
- La integración es la actividad en la cual se combinan los componentes de software individuales en subsistemas más amplios.
- La integración adicional de subsistemas también es parte del proceso de integración del sistema.
- Cada componente ya ha sido probado en lo referente a su funcionalidad interna (prueba de componente). Las pruebas de integración comprueban las funciones externas.
- Puede ser ejecutada por los desarrolladores y/o los testers

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Alcance /1

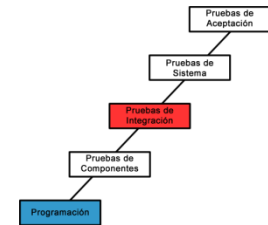


- Las pruebas de integración asumen que los componentes ya han sido probados.
- Las pruebas de integración comprueban la interacción mutua entre componentes (subsistemas) de software:
 - Interfaces con otros componentes.
 - Interfaces GUIs / MMIs.
- Las pruebas de integración comprueban las interfaces con el entorno del sistema.
 - En la mayoría de los casos la Interacción probada es el comportamiento del componente y el entorno simulado.
 - En condiciones reales factores adicionales del entorno pueden influir en el comportamiento del componente. Por lo tanto las pruebas de integración no pueden garantizar un comportamiento correcto en el entorno real del sistema.
- Los casos de prueba pueden ser derivados de la especificación de interfaz, diseño estructural o modelos de datos

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Alcance /2

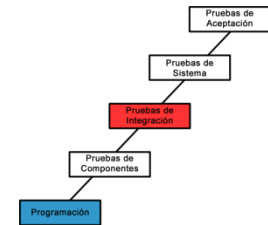


- Será probado un (subsistema) sistema compuesto de componentes Individuales.
 - Cada componente tiene una Interfaz externa y/o una interfaz que interactúa con otro componente dentro del mismo (subsistema) sistema.
- Son necesarios **DRIVERS** de prueba (los cuales aportan el entorno al proceso del sistema o subsistema)
 - Con el objeto de tener en cuenta o producir entradas y salidas del (subsistema) sistema.
 - Con el objeto de registrar datos.
- Los DRIVERS de prueba propios de las pruebas de componente pueden ser reutilizadas en estas pruebas.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Alcance /3

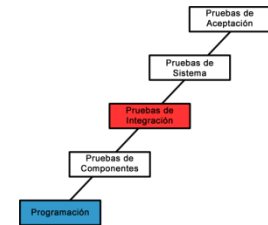


- Herramientas de control (monitoring tools): pueden apoyar las actividades de pruebas registrando datos y controlando las mismas pruebas.
- Un **STUB** reemplaza un componente faltante.
 - Un **STUB** programado reemplazará datos o funcionalidad de un componente que aún no ha sido integrado.
 - Un **STUB** asumirá las tareas elementales de un componente faltante.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Enfoque

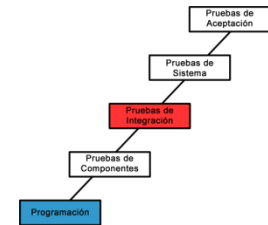


- El propósito de las pruebas de Integración es detectar defectos en las interfaces. Las pruebas de integración comprueban la correcta interacción entre componentes.
 - Con el objeto de comprobar, entre otros, aspectos relativos al rendimiento y la seguridad, requiriendo pruebas adicionales (no funcionales).
- Al reemplazar **DRIVERS** y **STUBS** de prueba por componentes reales se pueden generar nuevos defectos tales como:
 - Pérdida de datos, manipulación errónea de datos o entradas erróneas.
 - El componente involucrado interpreta los datos de entrada de una manera distinta.
 - Los datos son transferidos en un momento incorrecto: antes de tiempo, después de tiempo, a una frecuencia distinta de la requerida.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Estrategias /1

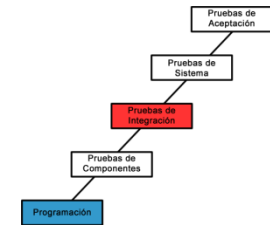


- Hay diferentes estrategias para las pruebas de Integración.
 - El enfoque Incremental es un elemento común a la mayoría de las estrategias (excepción: estrategia BIG BANG).
 - Las estrategias ascendentes (BOTTOM-UP) y descendente (TOP-DOWN) son los utilizados con mayor frecuencia.
- La elección de la estrategia debe considerar también aquellos aspectos relativos a la eficiencia de las pruebas.
 - La estrategia de integración determina la magnitud del esfuerzo requerido para las pruebas (por ejemplo el uso de herramientas, programación de **DRIVERS** y **STUBS**, etc.).
 - La finalización de la construcción de componentes determina, para todos los tipos de estrategias, el intervalo temporal en el cual el componente estará disponible. Por lo tanto, la estrategia de desarrollo influye en la estrategia de integración.
- En cada proyecto específico se debe considerar el compromiso entre la reducción de tiempo y la reducción de esfuerzo en pruebas:
 - Probar aquello que se encuentra finalizado: mayores costos en pruebas y menor tiempo ocioso.
 - Seguir un plan de pruebas de integración estricto: menores costos y mayor tiempo ocioso.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Estrategias /2

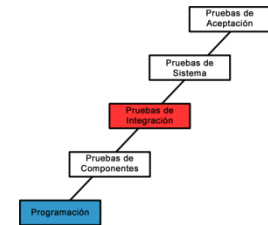


- Integración AD HOC.
 - Los componentes serán probados, si es posible, inmediatamente después de haber sido finalizada su construcción y se hayan finalizado las pruebas de componente.
- Características de la integración AD HOC.
 - Inicio temprano de las actividades de prueba, dando lugar a un proceso de desarrollo de software más corto en términos globales.
 - DRIVERS y STUBS serán necesarios dependiendo del tipo de componentes finalizados.
- Uso de la Integración AD HOC.
 - Es una estrategia que puede ser aplicada en cualquier etapa del proyecto.
 - Es una estrategia que normalmente se aplica combinada con otras.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de integración - Resumen

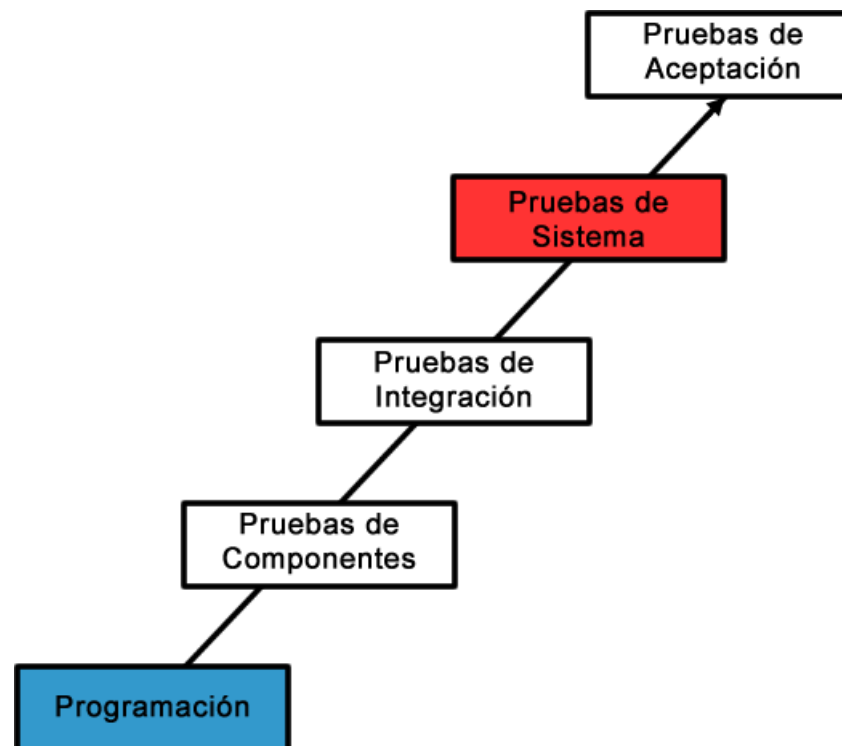


- Integración significa construir grupos de componentes.
- Las pruebas de integración comprueban la interacción entre componentes respecto de la especificación de interfaces.
- La integración ocurre de forma ascendente (BOTTOM-UP), descendente (TOP-DOWN) o en forma de BIG BANG.
- Integración de subsistemas (están constituidos por componentes integrados) también es un forma de integración.
- Cualquier estrategia de integración distinta a las anteriores es integración AD HOC.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

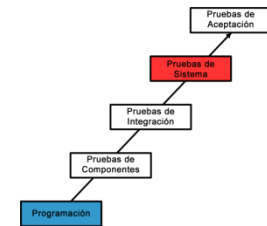
Pruebas de Sistema



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema

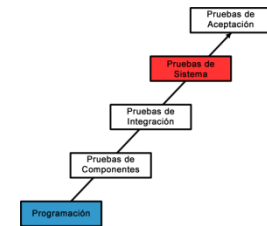


- Base de pruebas:
 - Especificación de requerimientos del sistema y del software
 - Casos de uso
 - Especificación funcional
 - Informes de análisis de riesgo
 - Manuales de operación del sistema y de usuario
 - Configuración del sistema
- Parametrización (Datos de configuración)

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema - Definición



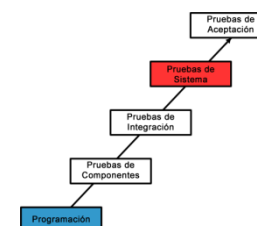
- El proceso de pruebas de un sistema integrado comprueba si cumple con los requisitos especificados.
- Las pruebas del sistema verifica el comportamiento de todo el sistema
- El alcance de las pruebas se define en el Plan Maestro de pruebas o en el Plan de niveles de pruebas
- Se busca que la calidad del software este determinado por el punto de vista del usuario
- Pruebas del sistema se refieren (según ISO 9126)
 - Requisitos funcionales (funcionalidad y no funcionales (fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad)
- Las características de la calidad de los datos son las bases para las pruebas
- Los casos de prueba se pueden derivar de:
 - Especificaciones funcionales
 - Casos de uso
 - Procesos de negocio
 - Informes de análisis de riesgo

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema – Alcance

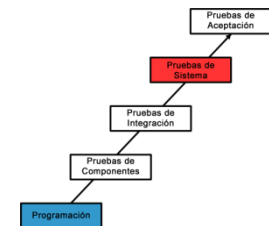
- Pruebas de un sistema integrado desde el punto de vista del usuario.
 - Implementación completa y correcta de los requerimientos.
 - Despliegue en el entorno real del sistema con datos reales.
- El entorno de pruebas debería coincidir con el entorno real.
 - No son necesarios STUBS o DRIVERS.
 - Todas las interfaces externas son probadas en condiciones reales.
 - Emulación próxima al futuro entorno real del sistema
- **No se realizan pruebas en el entorno real**
 - Los defectos inducidos podrían dañar el entorno real.
 - Un software objeto de despliegue se encuentra en un estado de cambio constante.
Muchas pruebas no serán reproducibles.



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema – Requerimientos funcionales /1

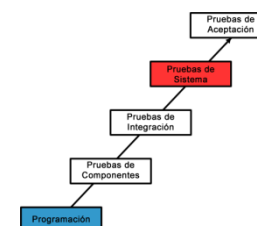


- **Objetivo:** comprobar que la funcionalidad implementada expone las características requeridas.
- Las características a ser probadas incluyen (según ISO 9126):
 - **Adecuación/idoneidad (Suitability).**
 - ¿Las funciones Implementadas son adecuadas para su uso esperado?
 - **Exactitud (Accuracy).**
 - ¿Las funciones presentan los resultados correctos (acordados)?
 - **Interoperabilidad (Interoperability).**
 - ¿Las interacciones con el entorno del sistema presentan algún problema?
 - **Conformidad (Compliance).**
 - ¿El sistema cumple con normas y reglamentos aplicables?
 - **Seguridad (security).**
 - ¿Están protegidos los datos/programas contra acceso no deseado o pérdida?

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema – Requerimientos funcionales /2

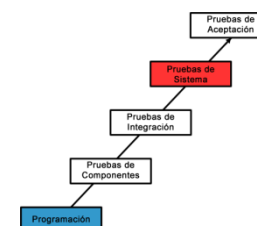


- Tres enfoques para probar requisitos funcionales:
 - **Pruebas basadas en requerimientos.**
 - Los casos de prueba se derivan de la especificación de requerimientos.
 - El número de casos de prueba variará en función del tipo/profundidad de las pruebas basadas en la especificación de requerimientos.
 - **Pruebas basadas en procesos de negocio.**
 - Cada proceso de negocio sirve como insumo para derivar/generar pruebas.
 - El orden de relevancia de los procesos de negocio pueden ser aplicados para asignar prioridades a los casos de prueba.
 - **Pruebas basadas en casos de uso:**
 - Los casos de prueba se derivan/generan a partir de las secuencias de usos esperados o razonables.
 - Las secuencias utilizadas con mayor frecuencia reciben una prioridad más alta.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema – Requerimientos no funcionales

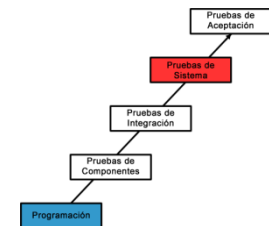


- La conformidad con requisitos no funcionales es difícil de lograr:
 - Con frecuencia la definición de estos requisitos es muy vaga (por ejemplo fácil de operar, interfaz de usuario bien estructurada, etc.).
 - Los requisitos no funcionales no son establecidos de forma explícita, son una parte implícita de la descripción del sistema. Sin embargo se espera que sean satisfechos.
 - La cuantificación de requisitos no funcionales es difícil debido al hecho de que se dispone de pocas o ninguna métrica objetiva para su medición: Las personas exponen su punto de vista subjetivo (por ejemplo que sea bonito, muy seguro, fácil de aprender, etc.).
- Ejemplo: Prueba / inspección de documentación.
 - ¿Está actualizada la documentación de programas con la versión actual del sistema?
 - ¿La documentación es completa, concisa y fácil de entender?
- Ejemplo: Prueba de mantenibilidad.
 - ¿Los programadores han cumplido con los estándares de codificación respectivos?
 - ¿El sistema ha sido diseñado de una forma estructurada y modular?

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Sistema – Resumen

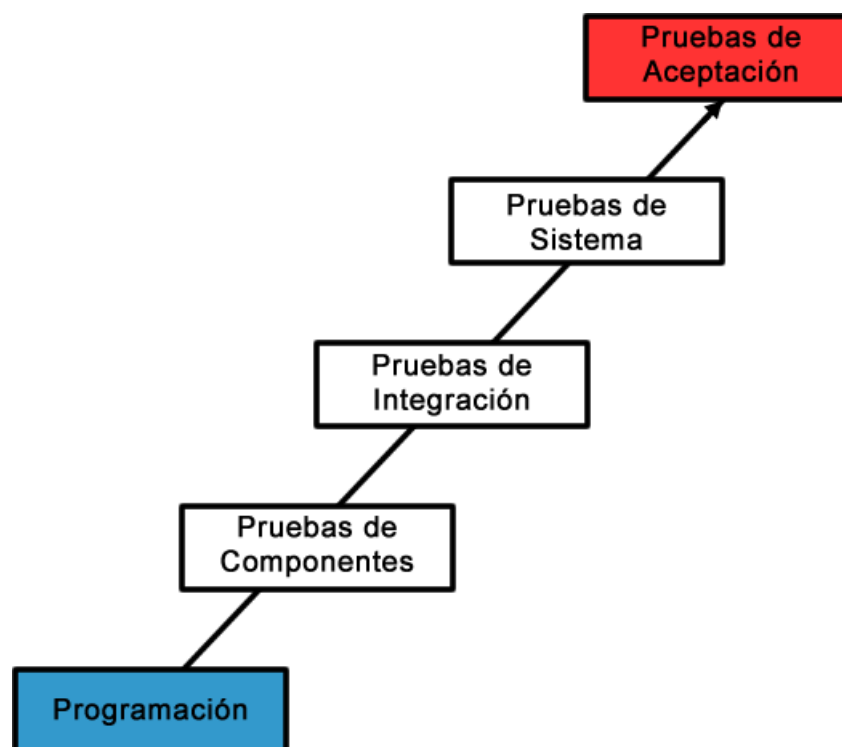


- Las pruebas de sistema se desarrollan utilizando casos de prueba funcionales y no funcionales.
- Las pruebas de sistema funcionales confirman que los requisitos para un uso específico previsto han sido satisfechos (validación).
- Las pruebas de sistema no funcionales verifican los atributos de calidad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

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

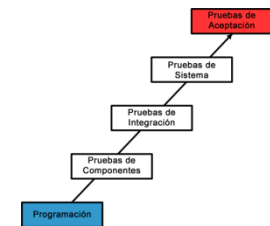
Pruebas de Aceptación



II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación - Definición

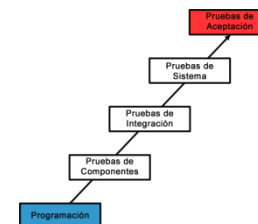


- Base de Pruebas:
 - Requerimientos de usuarios
 - Requerimientos del sistema
 - Casos de uso
 - Procesos de negocio
 - Informes de análisis de riesgo
- Objetos típicos de prueba:
 - Procesos de negocio totalmente integrados en el sistema
 - Procesos operativos y de mantenimiento
 - Procedimientos de usuario
 - Formularios
 - Informes
- Parametrización (Datos de configuración)

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación – Aceptación contractual

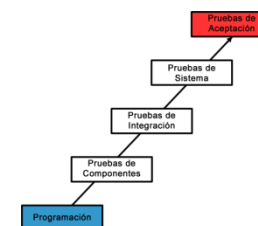


- ¿El software satisface todos los requisitos contractuales?
 - Con la aceptación formal se cumplen los hitos legales: comienzo de fase de garantía, hitos de abono (pago), acuerdos de mantenimiento, etc.
 - Criterios de aceptación verificables definidos en el momento del acuerdo contractual constituyen una garantía para ambas partes.
 - Las pruebas de aceptación deben tener en cuenta normas y reglamentos gubernamentales, legales, industriales y de otro tipo.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación – Aceptación de usuario

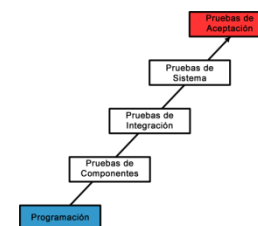


- Normalmente el cliente selecciona casos de prueba para las pruebas de aceptación.
 - Es posible que surjan malas Interpretaciones con respecto a los requerimientos y pueden ser discutidos. "El cliente conoce su negocio".
- Las pruebas se realizan utilizando el entorno del cliente.
 - El entorno del cliente puede producir nuevos fallas.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación – Aceptación operacional

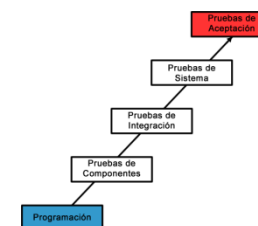


- Requiere que el software sea apto para su uso en el entorno productivo
 - Integración de software en la infraestructura IT (Information Technology) del cliente (Copia de seguridad, sistemas de recuperación, etc.)
 - Administración de usuarios, interfaz con las estructuras de archivos y directorios en uso
 - Compatible con otros sistemas (otros ordenadores, servidores de bases de datos, etc.)
 - Tareas de mantenimiento
 - Datos de carga y tareas de migración
 - Controles periódicos de vulnerabilidades del sistema
- Pruebas de aceptación operacional se hacen a menudo por el administrador del sistema del cliente

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación – Pruebas ALPHA Y BETA

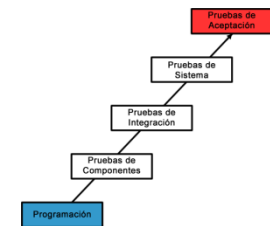


- Una versión preliminar y estable del software (versión BETA) se entrega a un conjunto específico (seleccionado) de clientes.
 - El cliente ejecuta las pruebas en dependencias propias (pruebas BETA, también denominadas pruebas de campo), los problemas son registrados e Informados o los desarrolladores para su corrección.
 - El cliente utiliza el software para hacer el tratamiento de sus procesos cotidianos o ejecuta un juego de pruebas seleccionado.
 - El software se prueba mejor en los distintos entornos del cliente.
 - Previo a las pruebas BETA él cliente pueden probar el nuevo software en las dependencias del desarrollo.
- Las pruebas ALPHA se desarrollan en las dependencias de lo organización que desarrolla.
- Ventajas de estas pruebas.
 - Reducen el costo de los pruebas de aceptación
 - Se utilizan distintos entornos de cliente
 - Involucran a un alto número de usuarios.

II - Pruebas a lo largo del ciclo de vida software

02 – Niveles de prueba.

Pruebas de Aceptación – Resumen



- 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 requerimientos del cliente.
- Las pruebas ALPHA y BETA son pruebas ejecutadas por clientes reales o potenciales
- Las Pruebas ALPHA se ejecutan en las dependencias del desarrollador
- Las pruebas BETA se ejecutan en las dependencias del cliente

II - Pruebas a lo largo del ciclo de vida software

Agenda

Capítulo II - Pruebas a lo largo del ciclo de vida software

- II/01 Modelos de desarrollo software.
- II/02 Niveles de prueba.
- **II/03 Tipos de prueba - objetivos de las pruebas.**
- II/04 Pruebas de Mantenimiento

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Tipos de pruebas y niveles de pruebas

- **Niveles de pruebas.**

- En la sección previa se han explicado los distintos niveles de pruebas, es decir pruebas de componente, pruebas de integración, etc.
- (En cada nivel de prueba los objetivos de las pruebas tienen un enfoque diferente
- Por lo tanto, se aplican distintos tipos de pruebas durante los distintos niveles de pruebas.

- **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 relacionadas con los cambios | Objetivo: probar después de cambios. |

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas funcionales

- **Objetivo: Validar la funcionalidad del objeto de prueba.**
 - La funcionalidad puede ser vinculada a los datos de entrada y salida de un objeto de prueba.
 - Los métodos de caja negra (black box) se utilizan en el diseño de casos de prueba relevantes.
 - Las pruebas se desarrollan teniendo en cuenta los requisitos funcionales establecidos en las especificaciones: Conceptos, casos de estudio, reglas de negocio o documentos relacionados.
 - **Ámbito de aplicación.**
 - Los pruebas funcionales se pueden llevar a cabo en todos los niveles de prueba
 - **Ejecución.**
 - El objeto de prueba es ejecutado utilizando combinaciones de datos de prueba derivados a partir de los casos de prueba.
 - Los resultados de la ejecución de la prueba son comparados con los resultados esperados
 - **Pruebas de Seguridad**
 - Tipo de pruebas funcionales que examina elementos externos de la aplicación
 - Ataques maliciosos pueden dañar programas y datos
-

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas no funcionales

- **Objetivo: Validar las características del producto software.**
 - ¿De qué forma el software lleva a cabo la función?
 - Valida características no funcionales del software (Según ISO 9126: Fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad)
 - A menudo las características no funcionales a menudo son vagas, incompletas o inexistentes, dificultando las pruebas asociadas a las mismas.
- **Ámbito de aplicación.**
 - Las pruebas no funcionales se pueden llevar a cabo en todos los niveles.
 - Pruebas no funcionales típicas:
 - Pruebas de rendimiento (Performance testing), Pruebas de carga (Load testing), pruebas de estrés (Stress testing), pruebas de volumen (Volume testing)
 - Pruebas de las características de seguridad para el software (Testing of security features), pruebas de las características de seguridad del software (Testing of safety features).
 - Pruebas de estabilidad y robustez: (Stability and robustness testing)
 - Pruebas de compatibilidad (Compatibility testing)
 - Pruebas de usabilidad (Usability testing)
 - Pruebas de configuración (Configuration testing)
- **Ejecución.**
 - La conformidad con los requisitos no funcionales se miden utilizando requerimientos funcionales (seleccionados).

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas no funcionales (Pruebas de sistema) / 1

- **Prueba de carga (Load test).**
 - Sistema bajo carga (carga mínima, más usuarios/transacciones).
- **Prueba de rendimiento (Performance test).**
 - Rapidez con la cual un sistema ejecuta una determinada función.
- **Prueba de volumen (Volume test).**
 - Procesamiento de grandes cantidades de datos / Archivos.
- **Prueba de estrés (Stress test).**
 - Reacción a la sobrecarga / recuperación tras el retorno a un carga normal.
- **Prueba de estabilidad (Stability test).**
 - Rendimiento en "modo de operación continua".
- **Prueba de robustez (Test for robustness).**
 - Reacción a entradas erróneas o datos no especificados. Reacción a fallos hardware / recuperación ante situaciones de desastre.

II - Pruebas a lo largo del ciclo de vida software
03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas no funcionales (Pruebas de sistema) / 2

- **Pruebas de seguridad para el software (de datos) (Test for data) security).**
 - Protección contra accesos no autorizados.
 - Protección contra el robo y daño de datos.
- **Pruebas de cumplimiento (Compliance test).**
 - Cumplimiento de normas y regulaciones (Internos / externos).
- **Pruebas de usabilidad (Test for usability).**
 - Estructurado, comprensible, fácil de aprender para el usuario.
- **Otros aspectos no funcionales de la calidad:**
 - Portabilidad: Capacidad de ser remplazado, de ser Instalado, adaptabilidad
 - Mantenible: Verificable, estable, analizable, modificable.
 - Fiabilidad: Madurez, robustez, capacidad de recuperación.

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas estructurales

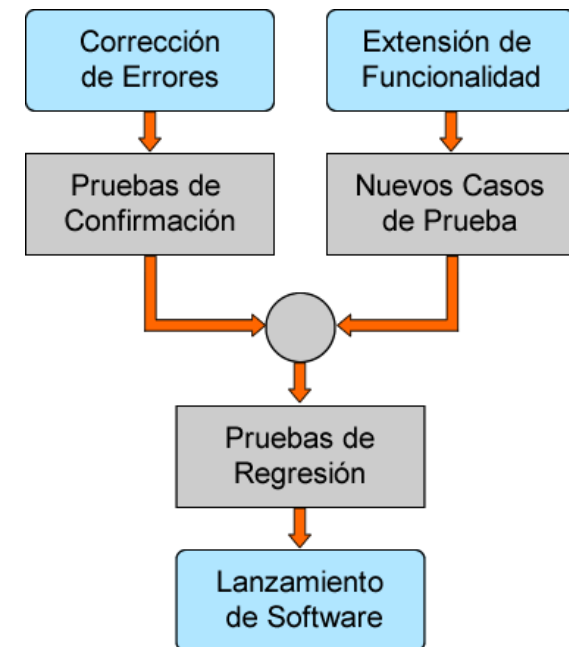
- **Objetivo: Validar cobertura de estructura**
 - Análisis de la estructura de un objeto de prueba (enfoque: caja blanca).
 - La finalidad de las pruebas es medir el grado en el cual la estructura del objeto de prueba ha sido cubierta por los casos de prueba.
- **Ámbito de aplicación.**
 - Las pruebas 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 finaliza tras haber sido diseñadas las pruebas funcionales con el propósito de obtener un alto grado de cobertura.
- **Ejecución.**
 - Se probará la estructura Interna de un objeto de prueba (por ejemplo el control de flujo en el Interior de un componente, el flujo a través de la estructura de un menú).
 - Objetivo: Todos los elementos estructurales identificados deben estar cubiertos por casos de prueba.

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas relacionadas con los cambios /1

- **Las pruebas relacionadas con los cambios son:**
 - **Pruebas de confirmación (re-testing):** Pruebas después de corrección de un defecto
 - **Pruebas de regresión:** Pruebas para descubrir nuevos defectos introducidos por un cambio.
- **Objetivo: Validar objeto de pruebas después de cambios.**
 - Dos razones principales
 - Corrección de defectos
 - Extensión o cambio de la funcionalidad
 - Después de que un objeto de pruebas o el entorno de su sistema ha sido objeto de modificación todos los resultados de pruebas resultan inválidos, por lo que los pruebas deben ser repetidas.
 - Es necesario volver o probar zonas adyacentes debido a efectos colaterales no deseados de una funcionalidad extendida o nueva.



II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas relacionadas con los cambios /2

- **Ámbito de aplicación.**
 - Las pruebas relacionadas con los cambios pueden ser ejecutadas en todos los niveles de pruebas.
 - Repetir una prueba de funcionalidad que ya había sido verificada se llama prueba de regresión.
 - El alcance de las pruebas de regresión depende del riesgo e impacto de la nueva funcionalidad implementada (extensión o corrección de defectos)
 - El análisis de riesgo se llama análisis de impacto

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Pruebas relacionadas con los cambios /2

- Ejecución
 - Básicamente lo ejecución tiene lugar de la misma forma en la cual se han ejecutado las pruebas en iteraciones previas.
 - En la mayoría de los casos, una prueba de regresión completa no es viable dado sus altos costos y duración.
 - Un alto grado de modularidad en el software permite reducir las pruebas de regresión.
 - Criterios para la selección de casos de prueba de regresión:
 - Casos de prueba de alta prioridad.
 - Probar solamente la funcionalidad estándar, saltarse casos y variaciones especiales.
 - Probar solamente la configuración utilizada con mayor frecuencia.
 - Probar solamente subsistemas / zonas seleccionadas del objeto de pruebas
 - Si durante fases tempranas del proyecto resulta evidente que ciertas pruebas son adecuadas para las pruebas de regresión, se deberá considerar la automatización de estas pruebas.

II - Pruebas a lo largo del ciclo de vida software

03 – Tipos de prueba: Objetivos de las pruebas.

Resumen

- En niveles de pruebas distintos se utilizan diferentes tipos de pruebas.
- Los tipos de pruebas son: funcionales, no funcionales. estructurales y pruebas relacionadas con los cambios.
- Las pruebas funcionales comprueban el comportamiento de entradas y salidas de un objeto de pruebas.
- Las pruebas no funcionales comprueban las características de un producto
- Las pruebas no funcionales incluyen, pero no están limitadas a: pruebas de carga, pruebas de estrés, pruebas de rendimiento, pruebas de robustez.
- Las pruebas estructurales habituales son pruebas que comprueban el flujo de control y de datos midiendo la cobertura en el objeto de prueba.
- Pruebas importantes después de un cambio: pruebas de confirmación (retest) y pruebas de regresión.

II - Pruebas a lo largo del ciclo de vida software

Agenda

Capítulo II - Pruebas a lo largo del ciclo de vida software

- II/01 Modelos de desarrollo software.
- II/02 Niveles de prueba.
- II/03 Tipos de prueba - objetivos de las pruebas.
- **II/04 Pruebas de Mantenimiento**

II - Pruebas a lo largo del ciclo de vida software

04 – Pruebas de Mantenimiento

Pruebas después de la aceptación del producto /1

- El cliente ha aprobado el producto y es puesto en producción.
 - El ciclo de desarrollo inicial, incluidas las pruebas asociadas, ha sido completado.
- El mismo software se encuentra al comienzo del ciclo de vida:
 - Será utilizado por muchos años, será ampliado.
 - Es muy probable que aún contenga defectos, por lo tanto será modificado y corregido.
 - Necesitará adaptarse a nuevas condiciones y deberá integrarse a nuevos entornos.
 - Necesitará cambiar o extender la parametrización (datos de configuración).
 - El software será retirado, puesto fuera de servicio.
- **Cualquier nueva versión del producto, cada nueva actualización y cada cambio del software requiere pruebas adicionales.**

II - Pruebas a lo largo del ciclo de vida software

04 – Pruebas de Mantenimiento

Pruebas después de la aceptación del producto /2

- **Configuración:**
 - La composición de un componente o sistema está definido como el número, naturaleza e interconexiones de sus partes.
- **Análisis de Impacto**
 - Se realiza evaluación de impacto basándose en la documentación de despliegue, documentación de pruebas y componentes, para implementar un cambio de un requerimiento específico
- **Pruebas de Mantenimiento**
 - Se prueban los cambios de un sistema que se encuentra operativo o el impacto de un cambio en el ambiente del sistema que se encuentra operativo

II - Pruebas a lo largo del ciclo de vida software

04 – Pruebas de Mantenimiento

Pruebas después de la aceptación del producto /3

- **El mantenimiento de software cubre dos diferentes campos:**
 - Mantenimiento como tal: Corrección de defectos o implementación de parches (HOT-FIXES), que eran parte de la versión inicial del software
 - Versiones planificadas del sistema: Adaptación como resultado de un cambio en el ambiente o nuevos requerimientos de los clientes

- **Alcance de las pruebas de mantenimiento**
 - Parches y corrección de defectos requieren pruebas de confirmación (retest)
 - Extensión de la funcionalidad requiere de nuevos casos de prueba
 - Migración a otras plataformas requiere de pruebas operacionales
 - Adicionalmente, intensivas pruebas de regresión son necesarias

II - Pruebas a lo largo del ciclo de vida software

04 – Pruebas de Mantenimiento

Pruebas después de la aceptación del producto /4

- El alcance de las pruebas está determinado por el impacto del cambio
 - El análisis de impacto es usado para determinar las áreas afectadas para decidir la cantidad de pruebas de regresión
 - Pueden ocurrir problemas si la documentación del software antiguo es incorrecta o incompleta

- Retiro del software
 - Las pruebas después de que el software es retirado puede incluir
 - Pruebas de migración de datos
 - Verificar archivado de datos y de programas
 - Pruebas en paralelo del antiguo y el nuevos sistema

II - Pruebas a lo largo del ciclo de vida software

04 – Pruebas de Mantenimiento

Pruebas después de la aceptación del producto /4

Resumen

- El software desarrollado y que se encuentra operativo, necesita ser adaptado a nuevas condiciones, los errores deben ser corregidos
- Un análisis de impacto puede ayudar a determinar los riesgos relacionados con los cambios
- Las pruebas de mantenimiento aseguran que:
 - Nuevas funcionalidades ha sido implementadas correctamente (nuevos casos de uso)
 - Errores han sido corregidos (antiguos casos de uso)
 - La funcionalidad ya ha sido verificada y no fue afectada (pruebas de regresión)
 - Si el software se retira, pruebas de migración o pruebas en paralelo pueden ser necesarias

III – Técnicas Estáticas

Agenda

Capítulo III – Técnicas Estáticas

- **III/01 Técnicas Estáticas y el proceso de pruebas**
- III/02 Proceso de Revisión
- III/03 Análisis Estático por Herramientas

III – Técnicas Estáticas

01 – Técnicas Estáticas y el proceso de pruebas

Enfoque Básico

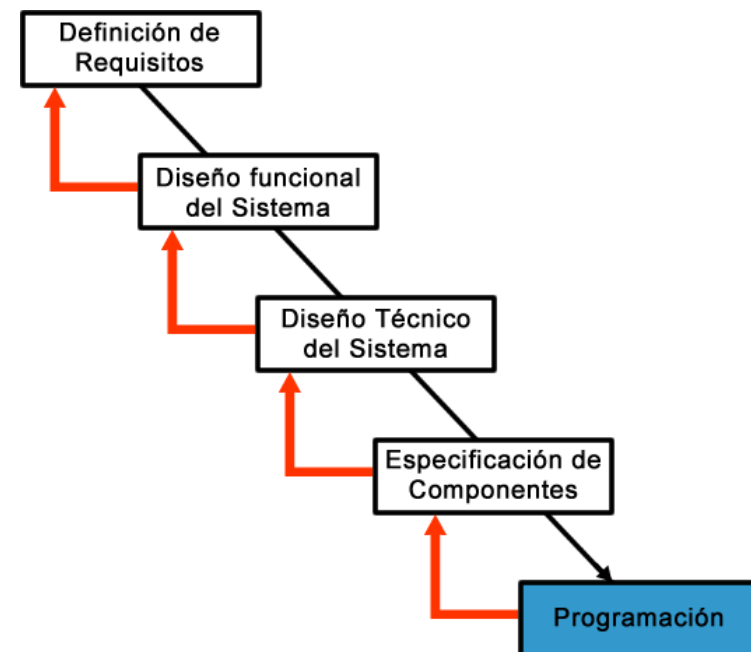
- Las técnicas estáticas de pruebas comprenden varios métodos que no ejecutan el componente o sistema objeto de la prueba.
- Las pruebas estáticas incluyen:
 - Revisiones (Actividad manual).
 - Análisis estático (Actividad basada en herramientas).
- Las técnicas estáticas complementan los métodos dinámicos.
 - Las pruebas estáticas detectan defectos en lugar de fallos.
 - Los conceptos también son analizados, no sólo el código ejecutable.
 - Los defectos / desviaciones son detectados en una fase temprana, antes de que el código sea implementado
 - Las pruebas estáticas encuentran defectos que no son posibles de encontrar con las pruebas dinámicas
- Documentos de alta calidad conducen a productos de alta calidad.
 - Incluso si una especificación revisada no contiene ningún defecto, la interpretación de la especificación y creación del diseño pueden ser defectuosas

III – Técnicas Estáticas

01 – Técnicas Estáticas y el proceso de pruebas

Objetivos de las Revisiones

- Las revisiones se realizan con el objeto mejorar la calidad del producto.
 - Las revisiones se utilizan para verificar la correcta transición de una fase a la siguiente, según está definido en el lado izquierdo del modelo-V.
- La detección temprana de errores ahorra costos.
- En el transcurso de las revisiones se podrían detectar los siguientes defectos:
 - Defectos en las especificaciones.
 - Defectos en el diseño y arquitectura del software.
 - Defectos en las especificaciones de interfaces.
 - Insuficiencia de mantenibilidad (Código sin comentarios)
 - Desviaciones con respecto a estándares acordados



III – Técnicas Estáticas

01 – Técnicas Estáticas y el proceso de pruebas

Ventajas y Desventajas de las Revisiones

- **Ventajas.**

- Costos más bajos y ahorro potencial relativamente alto.
- Defectos en la documentación son detectados y corregidos tempranamente.
- Los documentos de alta calidad mejoran el proceso de desarrollo.
- Mejora el índice de comunicación / Intercambio de conocimiento (Know-how).

- **Desventajas.**

- Se podrían presentar situaciones de tensión por confrontaciones directas con el autor.
- Los expertos involucrados en las revisiones deben adquirir conocimientos específicos del producto. Es necesaria una buena preparación.
- Inversión considerable de tiempo (del 10% al 15% del presupuesto total).
- Moderador y participantes influyen directamente en la calidad de la revisión.

III – Técnicas Estáticas

Agenda

Capítulo III – Técnicas Estáticas

- III/01 Técnicas Estáticas y el proceso de pruebas
- **III/02 Proceso de Revisión**
- III/03 Análisis Estático por Herramientas

III – Técnicas Estáticas

02 – Proceso de Revisión

Introducción

- Las REVISIONES validan los productos derivados del desarrollo de software desde diferentes perspectivas.
- Las listas de verificación pueden hacer el proceso de revisión más eficiente.
- Una lista de verificación con problemas típicos pueden ayudar a encontrar incidencias (ISSUES) indetectables.

III – Técnicas Estáticas

02 – Proceso de Revisión

Actividades de una Revisión Formal /1

- **Planificación**
 - Definición de criterios de revisión (Listas de Chequeo, tipos de revisión)
 - Selección de personal (Revisores, moderadores, etc.)
 - Asignación de roles y tiempos en el cronograma de proyecto (quien hace que)
- **Definición de criterios de entrada y de salida para los tipos de revisión formal**
 - Definir que partes de los documentos se someten a revisión (depende de importancia y complejidad)
- **KICK-OFF**
 - Distribución de documentos
 - Explicar los objetivos, procesos y documentos
- **Verificar los criterios de entrada (para más tipos de revisión formal)**
- **Preparación individual**
 - Revisores inspeccionan objetos, toman notas de puntos/elementos a clarificar
- **Señalar posibles defectos, preguntas y comentarios**

III – Técnicas Estáticas

02 – Proceso de Revisión

Actividades de una Revisión Formal /2

- **Reunión de revisión**
 - Reunión de todos los participantes de la revisión, revisores presentan sus resultados
 - Discusión y registro con documentación de resultados
 - Se señalan defectos, se realizan recomendaciones para el manejo de defectos, se toman decisiones acerca de los defectos.
- **Examinar, Evaluar, Registrar**
 - Se envía al grupo de revisión comunicaciones electrónicas lo acontecido en las reuniones de seguimiento
- **REWORK**
 - Los autores corrigen los defectos encontrados por los revisores
 - Se registran los estados actualizados de los defectos
- **Seguimiento (FOLLOW UP)**
 - Verificar que los defectos se han abordado
 - Decidir si una segunda revisión es necesaria
- **Verificar criterios de salida para tipos de revisión formal.**
 - Se entrega el visto bueno al objeto de revisión

III – Técnicas Estáticas

02 – Proceso de Revisión

Roles y responsabilidades /1

- **Director - Responsable - Jefe de proyecto.**
 - Inicia la revisión, decide los participantes
 - Asigna tiempos y recursos en el cronograma de proyecto
 - Determina si los objetivos de la revisión se han cumplido

- **Moderador.**
 - Dirige la reunión/discusión, hace de mediador, concluye resultados.
 - Planifica y ejecuta las revisiones, realiza seguimiento posterior
 - De él depende el éxito de la revisión

- **Autor.**
 - Principal responsable del objeto de revisión
 - Expone su trabajo a la crítica, lleva a cabo los cambios recomendados.

III – Técnicas Estáticas

02 – Proceso de Revisión

Roles y responsabilidades /2

- **Revisor (Inspector, Evaluador).**

- Individuos con un conocimiento específico técnico o del negocio
- Detecta defectos, desviaciones, áreas problemáticas, etc.
- Deben estar presentes en cualquier reunión de revisión

- **Escritor (Escriba, Escribano).**

- Documenta todos los asuntos, problemas y puntos que hubieran sido identificados durante las reuniones de revisión

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /1

- El proceso básico de una revisión (como se esquematiza aquí) se aplica a las siguientes variaciones de las revisiones (tipos de revisión):
 - Inspección, Walkthrough, Revisión Técnica, Revisión Informal.
- Una diferencia adicional de las revisiones se presenta dependiendo de la naturaleza del objeto, producto o proceso de la revisión.
 - Proceso de desarrollo de Software o proceso del proyecto.
 - CMMI, IEC 12207 y TPI son términos relacionados con la mejora de procesos
 - También denominada revisión de gestión (management review). Estas revisiones no Interfieren directamente en el proceso de pruebas, no forman parte de este curso.
 - Documentos / productos del proceso de desarrollo.
 - Estas son las revisiones tratadas en el presente curso.

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /2

- **Inspección: Características Relevantes.**
 - Los revisores inspeccionan el objeto en revisión usando listas de comprobación (verificación) y métricas.
 - Un moderador capacitado (formación específica) e independiente dirige la revisión.
 - La viabilidad de la revisión del objeto es valorada de forma previa a la revisión.
 - Especifica criterios de entrada y de salida para aceptación del producto de software
 - Proceso formal basado en reglas y listas de comprobación para las actividades de preparación, ejecución, documentación y seguimiento.
 - Usualmente emplea examinación por pares
 - Requiere reuniones previas de preparación
 - El reporte de revisión incluye lista de resultados

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /3

- **Inspección: Ventajas y Desventajas.**
 - Sesiones formales y organizadas con roles claramente definidos.
 - Requiere actividades Intensas de preparación y seguimiento.
 - Son necesarios el moderador y el escriba.
 - Propósito principal: Encontrar defectos utilizando un método estructurado.

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /4

- **Walkthrough: Características Relevantes.**
 - Opcionalmente puede haber una reunión previa de preparación de los revisores.
 - Sesiones de inicio y finalización
 - Puede tomar la forma de escenarios, con examinación por pares
 - La reunión es dirigida por el autor que explica el objeto en revisión.
 - No es necesario un moderador distinto (el autor modera).
 - A lo largo de la presentación por parte del autor, los revisores tratan de localizar desviaciones y/o áreas que representen un problema.
 - **Ejemplos para el uso de Walkthroughs:**
 - Walkthrough de documentos.
 - Walkthrough de un diseño preliminar de interfaz de usuario.
 - Walkthrough de un modelo de datos del proceso de negocio.

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /5

- **Walkthrough: Ventajas y Desventajas.**
 - Esfuerzo reducido en la preparación de la sesión de revisión, pero es una sesión cuyo resultado puede quedar abierto.
 - Una sesión puede ser iniciada a través de notificaciones realizadas con poca antelación.
 - El autor tiene una gran influencia sobre el resultado: dado que él mismo modera la revisión, hay un riesgo de dominación por su parte (puntos críticos no abordados en profundidad).
 - Posibilidad limitada de control, dado que el autor también se encuentra a cargo de cualquier actividad de seguimiento.

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /6

- **Revisión técnica: Características relevantes /1**
 - La meta del examen es un aspecto técnico del objeto en revisión: ¿Es apto para el uso?
 - Son necesarios expertos, preferentemente externos con participación opcional del Director
 - Puede ser ejecutada como revisión por pares sin la participación del Director
 - Idealmente dirigida por un moderador capacitado que no sea el autor.

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /7

- **Revisión técnica: Características relevantes /2**
 - La reunión puede tener lugar sin la presencia del autor.
 - Requiere reuniones previas de preparación por parte de los revisores
 - Usa listas de verificación para no olvidar cosas importantes
 - Un panel de expertos presenta sus recomendaciones con carácter unánime.
 - Preparación de un reporte de revisión que incluye lista de hallazgos y el veredicto acerca de si el producto de software cumple los requerimientos
 - La revisión técnica puede ser formal o informal dependiendo de su importancia

III – Técnicas Estáticas
02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /8

- **Revisión técnica: Objetivos principales.**
 - Discusiones durante las revisiones
 - Toma de decisiones
 - Evaluación de alternativas
 - Encontrar defectos
 - Resolver problemas técnicos
 - Verificar conformidad con : estándares, planeación, especificaciones y regulaciones

III – Técnicas Estáticas

02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /9

- **Revisiones informales: Características relevantes.**
 - Es la forma de revisión más simple.
 - Frecuentemente Iniciada por el autor.
 - Solamente estarán involucrados los evaluadores (uno o más).
 - No es necesaria ninguna reunión.
 - Los resultados pueden ser registrados en forma de una lista de acción
 - Frecuentemente desarrolladas a través de la solicitud de revisión de un documento o un compañero de trabajo.
 - También denominada: revisión por pares (peer review).

III – Técnicas Estáticas
02 – Proceso de Revisión

Tipos de revisiones (IEEE 1028) /10

- **Revisiones informales: ventajas y desventajas.**
 - Fácil de ejecutar, incluso en los casos de notificaciones realizadas con poca antelación.
 - Rentable.
 - No requiere protocolo.

III – Técnicas Estáticas

02 – Proceso de Revisión

Factores de éxito de una revisión /1

- Las revisiones se deben desarrollar orientadas al logro de objetivos. Las desviaciones en el objeto revisado deben ser establecidas de forma Imparcial.
- El autor del objeto revisado deberá ser motivado de una forma positiva para la revisión ("su documento será aún mejor" en lugar de "su documento es de baja calidad").
- Uso sistemático (entre otras) de las técnicas y plantillas.
- El uso de listas de verificación mejorará la eficiencia de la revisión.
- Para la ejecución apropiada de las revisiones será necesario un presupuesto suficiente (10% al 15% del costo total del desarrollo).
- Hacer uso de las lecciones aprendidas, utilizar la retroalimentación para implementar un proceso de mejora continua.
- Los directores de proyecto pueden dar soporte para realizar un buen proceso de revisión

III – Técnicas Estáticas

02 – Proceso de Revisión

Factores de éxito de una revisión /2

- Las revisiones deben ser desarrolladas en un ambiente de confianza.
- Los resultados no deben ser usados para evaluar a los participantes.
- Los testers o probadores realizan las principales contribuciones al proceso de revisión y aprenden del producto para poder realizar la preparación de pruebas tempranas.
- Para cumplir los objetivos de las revisiones deben estar involucradas las personas correctas.
- Hay un énfasis en el aprendizaje y en el proceso de mejoras

III – Técnicas Estáticas

02 – Proceso de Revisión

Resumen

- En el transcurso de las pruebas estáticas no se ejecuta el objeto de prueba.
- Las revisiones pueden tener lugar en fases tempranas del proceso de desarrollo, ellas complementan/extienden los métodos de pruebas dinámicas.
- Fases de una revisión:
 - Planificación - preparación - preparación individual - reunión - rework - seguimiento.
- Roles y tareas para una revisión:
 - Director - moderador - autor - evaluador - escriba.
- Tipos de revisiones:
 - Inspección - walkthrough - revisión técnica - revisión informal.

III – Técnicas Estáticas

Agenda

Capítulo III – Técnicas Estáticas

- III/01 Técnicas Estáticas y el proceso de pruebas
- III/02 Proceso de Revisión
- **III/03 Análisis Estático por Herramientas**

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Terminología y definiciones

- **Análisis estático (Definición):**

Es aquella actividad que consiste en el análisis de un objeto de prueba. Por ejemplo código fuente, guión (script), requisito) llevado a cabo sin ejecutar el producto software.

- Posibles aspectos o ser comprobados con análisis estático:

- Reglas y estándares de programación.
- Diseño de un programa (análisis del flujo de control).
- Uso de datos (análisis del flujo de datos).
- Complejidad en la estructura del programa. (Métricas, por ejemplo número ciclomático)

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Aspectos generales /1

- Todos los objetos de prueba deben tener una estructura formal.
 - Esto es especialmente importante cuando se utilizan herramientas de pruebas.
 - Con mucha frecuencia no se generan documentos formalmente.
 - En la práctica, lenguajes de modelado, programación y de guión (scripting) cumplen con la regla, de la misma forma que algunos diagramas.
- El análisis estático de un programa mediante el uso de herramientas se desarrolla con un menor esfuerzo que el necesario en una revisión.
 - Frecuentemente el análisis estático se ejecuta antes de que tenga lugar una revisión.
 - Para encontrar un error lógico (potenciales loops infinitos)
 - Para descubrir construcciones complicadas, vulnerabilidades de seguridad, interfaces inconsistentes, etc.

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Aspectos generales /2

- El valor del análisis estático es la prevención de defectos
- Encontrar defectos tempranamente antes de la ejecución de pruebas
- Advertir de aspectos sospechosos en el código
- Detectar discrepancias en el diseño y para el cálculo de medidas de alta complejidad
- Detectar inconsistencias y dependencias
- Verificar la mantenibilidad del diseño del código
- Estos defectos no son fáciles de detectar con pruebas dinámicas

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Aspectos generales /3

Herramientas como compiladores y herramientas de análisis (analizador) pueden ser usadas

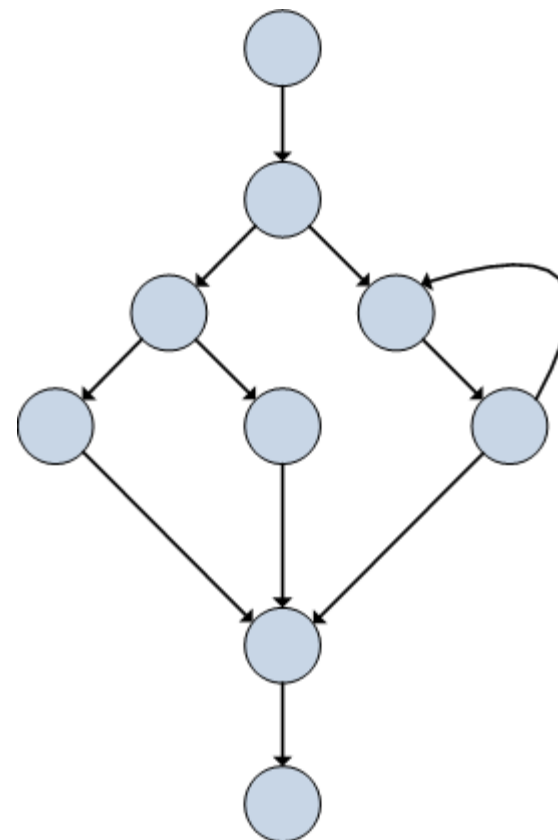
- **Compilador:**
 - Detecta errores de sintaxis en el código fuente de un programa.
 - Crea datos de referencia en el programa. Por ejemplo lista de referencias cruzadas, llamada Jerárquica, tabla de símbolos.
 - Comprueba la consistencia entre los tipos de variables.
 - Encuentra variables no declaradas y código Inaccesible (código muerto).
- **Analizador, trata aspectos adicionales tales como:**
 - Convenciones y estándares.
 - Métricas de complejidad.
 - Acoplamiento de objetos

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Análisis del flujo de control /1

- **Propósito.**
 - del código (ramas muertas, código muerto, etc.).
- **Método.**
 - La estructura del código se representa como un diagrama de control de flujo.
 - Grafo dirigido.
 - Los nodos representan sentencias o secuencias de sentencias.
 - Las aristas representan la transferencia del flujo de control, como decisiones y bucles.
 - Construcción mediante herramientas.

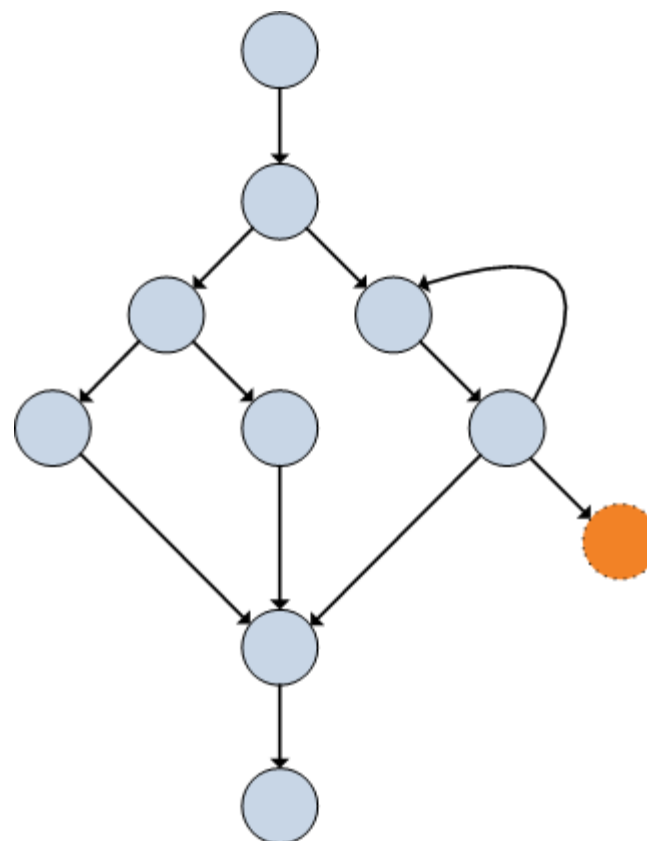


III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Análisis del flujo de control /2

- Resultados.
 - Mayor visión y comprensión del código del programa.
 - Las anomalías pueden ser fácilmente detectadas, los defectos se hacen evidentes.
 - Salida de un Bucles por un salto.
 - Ramas muertas.
 - Retornos múltiples.



III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Análisis del flujo de datos /1

- **Propósito.**

Detección de anomalías en el flujo de datos con la ayuda de los diagramas de control de flujo y conjeturas racionales con respecto de las secuencias del flujo de datos.

- **Beneficios.**

Detección fiable de anomalías en el flujo de datos. Se puede detectar fácilmente la localización exacta de defectos. Es un buen complemento para otros métodos de pruebas.

- **Desventajas.**

Limitado a un rango reducido de tipos de defectos.

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Análisis del flujo de datos /2

Método.

- Una variable x puede tener los siguientes estados a lo largo de la ejecución de un programa:
 - x se encuentra indefinida (**u**): no ha sido asignado valor alguno a la variable x .
 - x se encuentra definida (**d**): ha sido asignado un valor a x . Por ejemplo $x = 1$.
 - x está referenciada (**r**): ha sido tomada una referencia, el valor de x no cambia. Por ejemplo $x > 0$, $a = b + x$.
 - x no ha sido utilizada (**e**): x no ha sido referenciada (ni en lectura ni en escritura).
- El flujo de datos de una variable puede ser expresado como una secuencia de estados: **d, u, y r**. Por ejemplo $x \rightarrow u d r d r r u$
- Si una de estas secuencias contiene una sub-secuencia que no tiene sentido, entonces ha tenido lugar una anomalía en el flujo de datos.
 - Anomalía **ur**: Valores indefinidos han sido referenciados
 - Anomalía **du**: Definición de un valor y luego es indefinido antes de lectura
 - Anomalía **dd**: Definición de un valor y se define nuevamente antes de lectura

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Análisis del flujo de datos /3

Ejemplo de anomalía en el flujo de datos

Para este ejemplo, los valores de dos variables son intercambiados a través de una variable auxiliar, si no están ordenados por valor.

```
void MinMax (int Min, int Max)
{
    int Help;
    if (Min>Max)
    {
        Max = Help;
        Max = Min;
        Help = Min;
    }
    End MinMax;
```

El análisis de flujo de datos muestra:

- La Variable Help se encuentra indefinida (undefined) cuando es referenciada: **Anomalía ur.**
- La variable Max se define dos veces sin ninguna referencia entre ambas definiciones: **Anomalía dd.**
- El valor definido para la variable Help se vuelve indefinido al final del programa sin referencia: **Anomalía du.**

El ejemplo puede ser fácilmente corregido.

```
void MinMax (int Min, int Max)
{
    int Help;
    if (Min>Max)
    {
        Help = Max;
        Max = Min;
        Min = Help;
    }
    End MinMax;
```

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Las métricas y su cálculo

- Ciertos aspectos de la calidad de un programa pueden ser medidos utilizando métricas.
 - La métrica sólo tiene relevancia para el aspecto medido (considerado).
- La complejidad estática de un programa puede ser medida.
 - Actualmente hay aproximadamente 100 métricas diferentes disponibles.
- Métricas diferentes tratan aspectos diferentes de la complejidad de programa.
 - Tamaño del programa. Por ejemplo líneas de código.
 - Estructuras de control del programa. Por ejemplo número ciclomático.
 - Estructuras de control de datos. Por ejemplo métrica de Halstead
- **¡Es difícil comparar dos métricas diferentes, incluso cuando ambas abordan el mismo atributo del programa!**

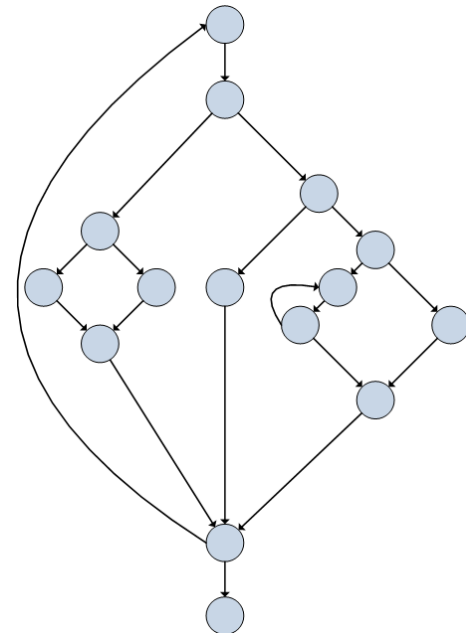
III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Las métricas y su implicación /1

Número Ciclomático: $v(G) = e - n + 2p$

- Métrica que mide la complejidad estática de un programa basada en su grafo de flujo de control.
- Mide los caminos linealmente independientes, como índice de la testabilidad y mantenibilidad.
- El número ciclomático se define de la siguiente forma:
 - Número de aristas: e.
 - Número de nodos: n.
 - Número de partes del programa Independientes Inspeccionadas: p (normalmente 1).
- Valores hasta 10 son aceptables. Para valores superiores el código debe ser mejorado (reworked - buena práctica según McCabe).



III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Las métricas y su implicación /2

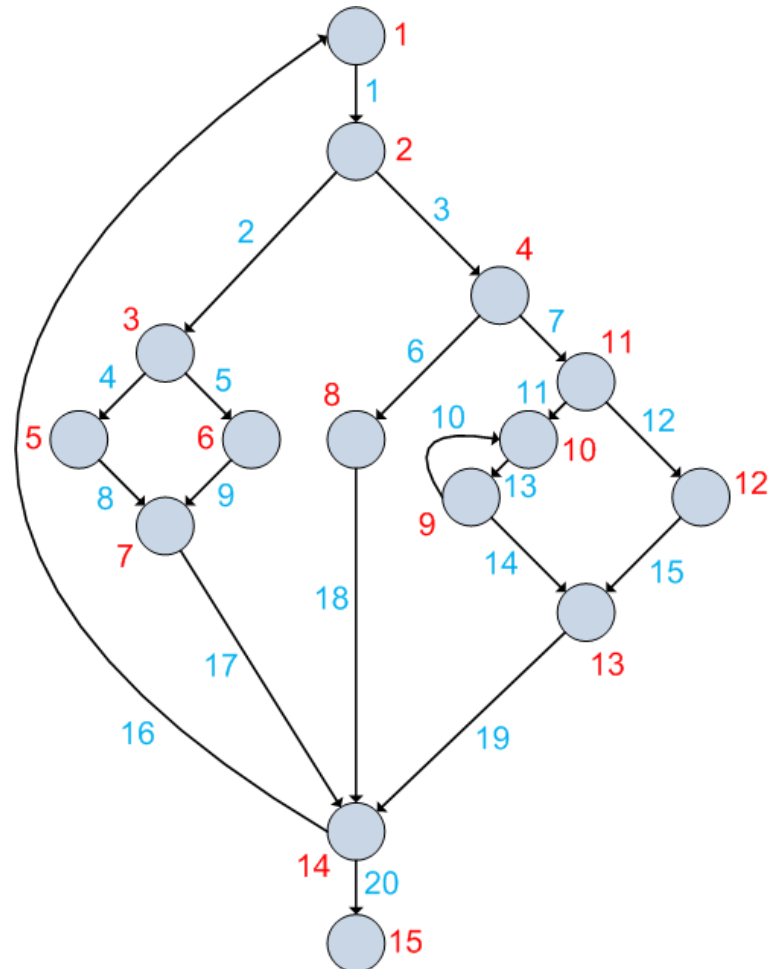
Ejemplo Número Ciclomático.

El grafo de la derecha tiene:

- 1 Partes independientes: $p = 1$
- 15 Nodos: $n = 15$
- 20 Aristas: $e = 20$

$$v(G) = e - n + 2p$$

$$v(G) = 7$$



III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Las métricas y su implicación /3

- **Número ciclomático (por McCabe) - implicación.**
- El número ciclomático puede ser utilizado como un valor objetivo para una revisión de código.
- El número ciclomático también puede ser calculado como el número de decisiones independientes más uno. Si las dos formas de cálculo aportan resultados diferentes se puede deber a:
 - Ramas superfluas
 - Ramas faltantes.
- El número ciclomático también aporta un índice del número de casos de prueba necesarios para alcanzar cobertura de decisión.

III – Técnicas Estáticas

03 Análisis Estático por Herramientas

Resumen

- **Análisis Estático.**

- Con el uso de herramientas para la realización de análisis estático (compiladores, analizadores) el código del programa puede ser objeto de Inspección sin ser ejecutado.
- Con el uso de herramientas se puede realizar el análisis estático de un programa con un esfuerzo menor que el necesario para una Inspección.

- **Resultado del Análisis.**

- El diagrama de control de flujo presenta el flujo del programa y permite la detección de ramas muertas y código inalcanzable.
- Las anomalías en los datos se detectan utilizando el análisis del flujo de datos.
- Las métricas pueden ser utilizadas para evaluar la complejidad estructural conduciendo o una estimación del esfuerzo en pruebas esperado.

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

- **IV/01 Pruebas en el proceso de desarrollo**
- IV/02 Categorías de las técnicas de diseño de pruebas.
- IV/03 Técnicas de caja negra (black box) o basadas en la especificación.
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- IV/05 Técnicas basadas en la experiencia.
- IV/06 Selección de las técnicas de pruebas.

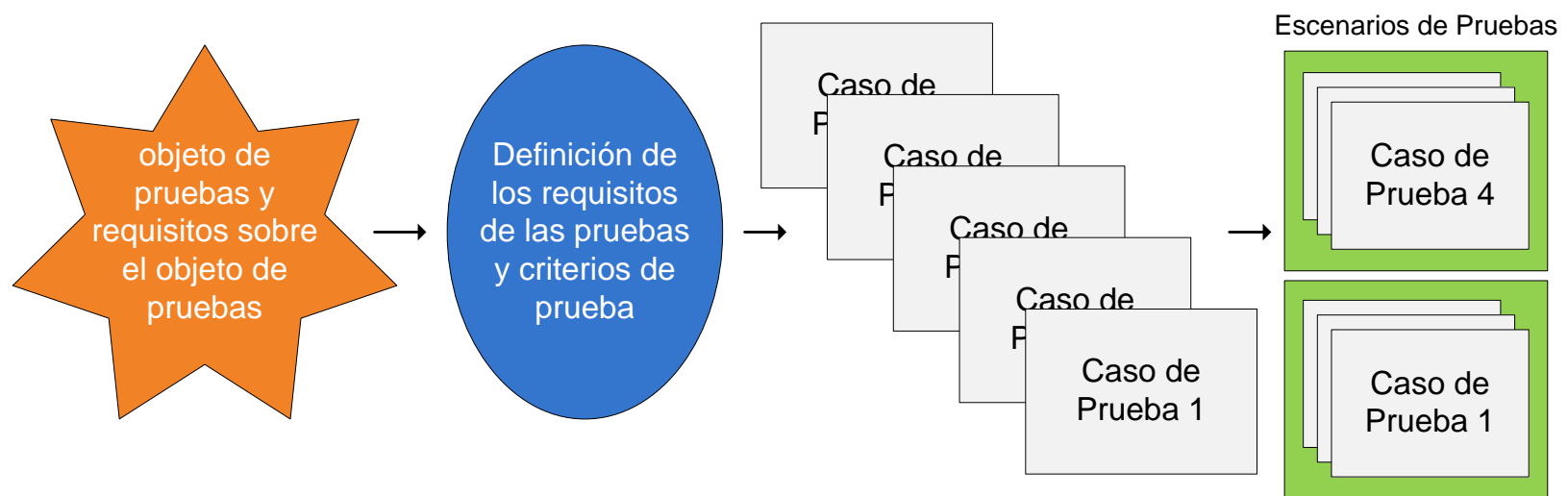
IV – Técnicas de Diseño de Pruebas

01 – Pruebas en el proceso de desarrollo

Obtención de casos de prueba a partir de requisitos

El diseño de casos de prueba debe ser un proceso controlado.

Los casos de prueba pueden ser creados formal o informalmente, dependiendo de las características del proyecto y la madurez del proceso en uso.



IV – Técnicas de Diseño de Pruebas

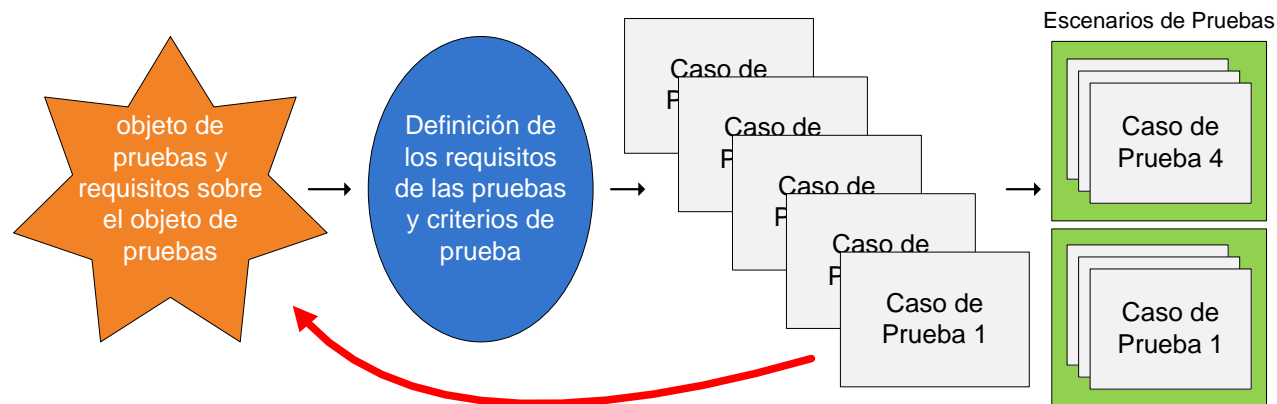
01 – Pruebas en el proceso de desarrollo

Trazabilidad

Las pruebas deben ser trazables: ¿Qué casos de prueba han sido incluidos en el catálogo de pruebas, basados en qué requisitos?

Las consecuencias de los cambios en los requerimientos sobre las pruebas a realizar se pueden identificar directamente

La trazabilidad también ayuda a determinar la cobertura de requisitos.



IV – Técnicas de Diseño de Pruebas

01 – Pruebas en el proceso de desarrollo

Definiciones

- **Objeto de pruebas (test object):**
 - El elemento o ser revisado: un documento o una pieza de software en el proceso de desarrollo de software.
 - **Condición de la prueba (test condition):**
 - Un elemento o evento: una función, una transacción, un criterio de calidad o un elemento en el sistema.
 - **Criterios de prueba (test criteria):**
 - El objeto de prueba debe cumplir los criterios de prueba con el objeto de superar la prueba.
 - **Cronograma de ejecución de pruebas (test execution schedule):**
 - Un cronograma para la ejecución de los procedimientos de pruebas. Los procedimientos son incluidos en el cronograma de ejecución de pruebas en su contexto y en el orden que debe ser ejecutado.
-

IV – Técnicas de Diseño de Pruebas

01 – Pruebas en el proceso de desarrollo

Descripción de un caso de prueba según el estándar IEEE 829:

- **Identificador único (distinguible) (distinct identification):** identificador o código único del caso de prueba.
- **Precondiciones (preconditions):** situación previa a la ejecución de pruebas o características de un objeto de pruebas antes de llevar a la práctica (ejecución) un caso de prueba.
- **Valores de entrada (Input values):** descripción de los datos de entrada de un objeto de pruebas.
- **Resultados esperados (expected results):** datos de salida que se espera produzca un objeto de pruebas
- **Poscondiciones (post conditions):** características de un objeto de pruebas tras la ejecución de pruebas, descripción de su situación tras la ejecución de las pruebas.
- **Dependencias (dependencies):** orden de ejecución de casos de prueba, razón de las dependencias.
- **Requisitos (requirements):** características del objeto de pruebas que el caso de prueba debe evaluar.

IV – Técnicas de Diseño de Pruebas

01 – Pruebas en el proceso de desarrollo

Combinación de casos de prueba

- Los casos de prueba se pueden combinar en set de pruebas (juegos de pruebas - test suites) y escenarios de pruebas.
- **Una especificación de procedimiento de prueba**, define la secuencia de acciones para la ejecución de un caso de prueba Individual o un set de pruebas. Es un guión o esquema de pruebas describiendo los pasos, el tratamiento y/o las actividades necesarias para la ejecución de pruebas.
- Los juegos de prueba pueden ser codificados y ejecutados de forma automática con el uso de herramientas adecuadas.
- **El plan de pruebas (test plan)**, establece la secuencia de los pruebas planificadas, quien debe ejecutarlas y cuándo. Las restricciones a considerar son las prioridades, la disponibilidad de recursos, la infraestructura de pruebas, etc.

IV – Técnicas de Diseño de Pruebas

01 – Pruebas en el proceso de desarrollo

Resumen

- Los casos de prueba y los set de pruebas (juegos de pruebas - test suites) son obtenidos a partir de los requisitos o características de los objetos de pruebas.
- Componentes de la descripción de un caso de prueba:
 - Código / identificador único.
 - Precondiciones (pre-conditions).
 - Valores de entrada (Input values).
 - Resultados esperados (expected results).
 - Poscondiciones (post-conditions).
 - Dependencias (dependencies).
 - Requisitos a partir de los cuales se ha obtenido el caso de prueba.

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

- IV/01 Pruebas en el proceso de desarrollo
- **IV/02 Categorías de las técnicas de diseño de pruebas.**
- IV/03 Técnicas de caja negra (black box) o basadas en la especificación.
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- IV/05 Técnicas basadas en la experiencia.
- IV/06 Selección de las técnicas de pruebas.

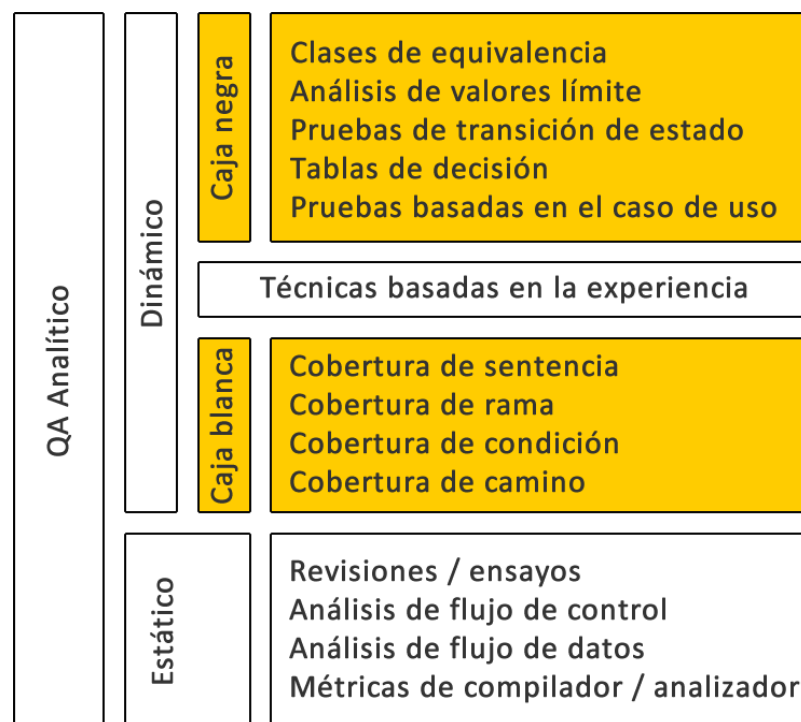
IV – Técnicas de Diseño de Pruebas

02 – Categorías de las técnicas de diseño de pruebas

Pruebas de caja negra (black box) y caja blanca (white box)

Las pruebas dinámicas se dividen en categorías / grupos:

- La agrupación se realiza en función del carácter básico del método utilizado para obtener los casos de prueba.
- Cada grupo tiene sus propios métodos para diseñar casos de prueba.

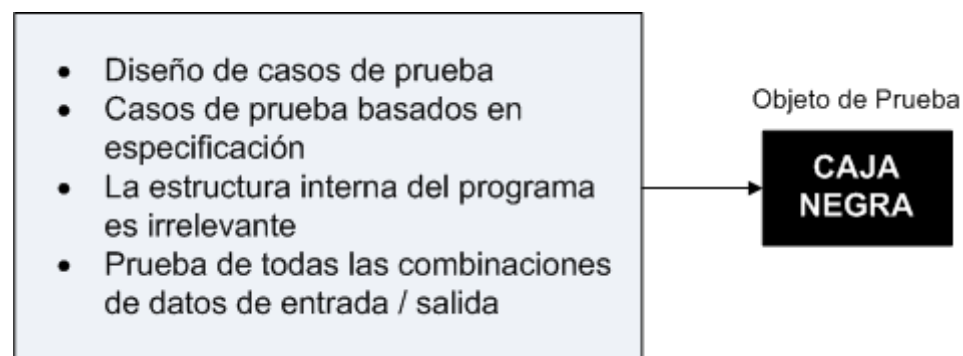


IV – Técnicas de Diseño de Pruebas

02 – Categorías de las técnicas de diseño de pruebas

Técnicas de caja negra (black box) o basadas en la especificación

- El probador observa el objeto de prueba como una caja negra.
 - La estructura Interna del objeto de prueba es irrelevante o desconocida.
- Los casos de prueba se obtienen a partir del análisis de la especificación (funcional y no funcional) de un componente o sistema.
 - Prueba del comportamiento entrada / salida (input / output).
- **La funcionalidad es el foco de atención**
 - La técnica de caja negra también se denomina **prueba funcional** o **prueba orientada a la especificación**.

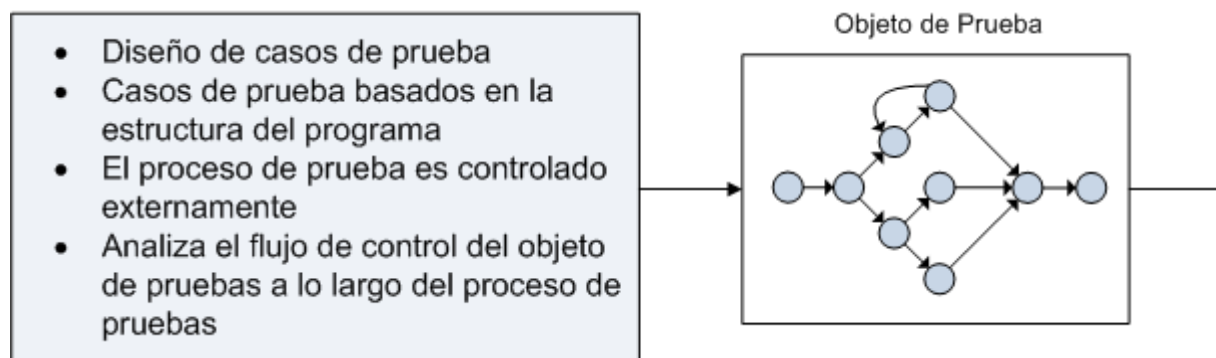


IV – Técnicas de Diseño de Pruebas

02 – Categorías de las técnicas de diseño de pruebas

Técnicas de caja blanca (white box) o basadas en la estructura

- El probador conoce la estructura Interna del programa / código.
 - Por ejemplo jerarquía de los componentes, flujo de control, flujo de datos, etc.
- Los casos de prueba son seleccionados en base a la estructura interna del programa / código.
 - A lo largo de las pruebas es posible que se interfiera con la ejecución de las pruebas.
- La estructura del programa es el foco de atención
 - La técnica de caja blanca también es conocida como **prueba estructural** o **pruebas basados en el flujo de control**.



IV – Técnicas de Diseño de Pruebas

02 – Categorías de las técnicas de diseño de pruebas

Categorías de las técnicas de diseño de pruebas - visión general

- **Métodos basados en la especificación.**
 - El objeto de prueba ha sido seleccionado de acuerdo con el modelo funcional de software.
 - La cobertura de la especificación puede ser medida (por ejemplo, el porcentaje de la especificación cubierta por casos de prueba).
- **Métodos basados en la estructura.**
 - La estructura Interna del objeto de prueba es utilizada para diseñar los casos de prueba (código/sentencias. menús, llamados, etc.).
 - El porcentaje de cobertura es medido y utilizado como fuente para la creación de casos de prueba adicionales.
- **Métodos basados en la experiencia.**
 - El conocimiento y experiencia respecto de los objetos de prueba y su entorno son las fuentes para el diseño de casos de prueba.
 - El conocimiento y experiencia respecto de posibles puntos débiles, posibles errores y errores previos son utilizados para determinar / definir casos de prueba.

IV – Técnicas de Diseño de Pruebas

02 – Categorías de las técnicas de diseño de pruebas

Resumen

- **Los casos de prueba pueden ser diseñados utilizando diferentes métodos.**
 - Si la funcionalidad especificada es el objetivo de las pruebas, los métodos utilizados se denominan métodos basados en la especificación o métodos de caja negra (black box).
 - Si la estructura interna de un objeto es Investigada, los métodos utilizados se denominan métodos basados en la estructura o métodos de caja blanca (white box).
 - Los métodos basados en la experiencia utilizan el conocimiento y la habilidad del personal involucrado en el diseño de casos de prueba.

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

- IV/01 Pruebas en el proceso de desarrollo
- IV/02 Categorías de las técnicas de diseño de pruebas.
- **IV/03 Técnicas de caja negra (black box) o basadas en la especificación.**
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- IV/05 Técnicas basadas en la experiencia.
- IV/06 Selección de las técnicas de pruebas.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box) o basadas en la especificación - Visión general

- En el presente apartado se explicarán en detalle los siguientes métodos de caja negra:
 - Partición de equivalencias o clases de equivalencia.
 - Análisis de valores límite.
 - Tablas de decisión y gráficos causa y efecto.
 - Pruebas de transición de estado.
 - Pruebas de caso de uso.
- Estos son los métodos más importantes y conocidos.
- Otros métodos de caja negra son:
 - Pruebas estadísticas.
 - Pruebas duales (algoritmo dual - pairwise).
 - Pruebas de humo.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

General

- Las pruebas funcionales están dirigidas a verificar la corrección y la completitud de una función.
 - ¿Están disponibles en el módulo todas las funciones especificadas?
 - ¿Las funciones ejecutadas presentan resultados correctos?
- La ejecución de casos de prueba debe tener una baja redundancia, sin embargo debe ser integral.
- Probar lo menos posible, probar tanto como sea necesario.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box)

- **Partición de equivalencias o clases de equivalencia.**
- Análisis de valores límite.
- Tablas de decisión y gráficos causa y efecto.
- Pruebas de transición de estado.
- Pruebas de caso de uso.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia

- La partición en clases de equivalencia es lo que la mayoría de los probadores hacen de forma Intuitiva. Dividen los posibles valores en clases, mediante lo cual observan.
 - Valores de entrada (Input valúes) de un programa (uso habitual del método CE).
 - Valores de salida (output valúes) de un programa (uso poco habitual del método CE)
- El rango de valores definido se agrupa en clases de equivalencia para las cuales se aplican las siguientes reglas:
 - Todos los valores para los cuales se espera que el programa tenga un comportamiento común se agrupan en una clase de equivalencia (CE)
 - Las clases de equivalencia pueden no superponerse y pueden no presentar ningún salto (discontinuidad)
 - Las clases de equivalencia pueden consistir en un rango de valores (por ejemplo $0 < x < 10$) o un valor aislado (por ejemplo $x = \text{Verdadero}$).

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Válida e Inválida

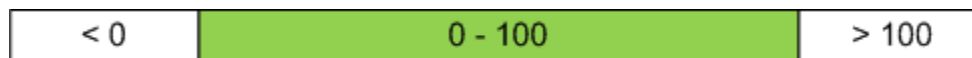
- Las clases de equivalencia de cada variable (elemento) se puede dividir en:
 - **CE Válida:** Todos los valores que están definidos en un rango combinado en una clase de equivalencia, si estos son manejados idénticamente por el objeto de pruebas
 - **CE Inválida:** Se distinguen dos casos para valores por fuera del rango definido:
 - Valores con un formato correcto pero por fuera del rango que puede ser combinado en una o más CE
 - Valores con un formato incorrecto son generalmente definidos en una CE separada
- Las pruebas se ejecutan utilizando un único representante de cada CE
 - Para todo otro valor perteneciente a la CE se espera el mismo comportamiento que para el valor seleccionado

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo

- Las clases de equivalencia se escogen para entradas (inputs) **válidas** e **inválidas**.
 - Si el valor x se define como $0 < x < 10$. entonces, inicialmente se pueden identificar tres clases de equivalencia:
 - $x < 0$ (valores de entrada no válidos)
 - $0 \leq x \leq 100$ (valores de entrada válidos)
 - $x > 100$ (valores de entrada no válidos)



- Se pueden definir CE adicionales, contenidas pero no limitadas a:
 - Entradas no numéricas
 - Números muy grandes o muy pequeños.
 - Formatos numéricos no admitidos

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Variables de entrada

- Todas las variables de entradas (input variables) del objeto de prueba son identificadas, por ejemplo:
 - Campos de una Interfaz Gráfica de Usuario (GUI)
 - Parámetros de una función.
- Se define un rango para cada valor de entrada (input).
 - Este rango define la suma de todas las clases de equivalencia válidas (CEv).
 - Las clases de equivalencia Inválidas (CEi) contienen los valores que no pertenecen al rango.
 - Aquellos valores que deben ser tratados de una forma diferente (conocidos o sospechosos) son asignados a una clase de equivalencia aparte.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Porcentajes /1

- Un programa espera el valor expresado en porcentaje de acuerdo a los siguientes requerimientos:
 - Solo enteros son permitidos
 - 0 es el valor inferior válido límite del rango
 - 100 es el valor superior límite del rango
- Válidos son: todos los números entre 0 y 100.
- Inválidos son: todos los números negativos, los mayores a 100, números decimales y valores no numéricos.

○ Una clase de equivalencia válida	(CEv)	$0 \leq x \leq 100$
○ Primera clase de equivalencia inválida	(CEi)	$x < 0$
○ Segunda clase de equivalencia inválida	(CEi)	$x > 100$
○ Tercera clase de equivalencia inválida	(CEi)	$x = \text{no entero}$
○ Cuarta clase de equivalencia inválida	(CEi)	$x = \text{no numérico}$

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Porcentajes /2

- El valor del porcentaje ahora se mostrará en un diagrama de barras. Los siguientes son requerimientos adicionales (los valores se incluyen en los rangos de cada CE)
 - Valores entre 0 y 15 barra gris
 - Valores entre 16 y 50 barra verde
 - Valores entre 51 y 85 barra amarilla
 - Valores entre 86 y 100 barra naranja

< 0	0 - 15	16 - 50	51 - 85	86 - 100	> 100
-----	--------	---------	---------	----------	-------

- Ahora son cuatro en lugar de una las clases de equivalencia válidas (CEv)
 - Primera clase de equivalencia válida (CEv) $0 \leq x \leq 15$
 - Segunda clase de equivalencia válida (CEv) $16 \leq x \leq 50$
 - Tercera clase de equivalencia válida (CEv) $51 \leq x \leq 85$
 - Cuarta clase de equivalencia válida (CEv) $86 \leq x \leq 100$

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Porcentajes /3

- **Elección de números representativos de cada CE**
 - Se determina un representante de cada CE y se define el resultado esperado para este.

Variable	Clase de equivalencia		Representativo
Valor de porcentaje VÁLIDO	CE ₁ :	$0 \leq x \leq 15$	+10
	CE ₂ :	$16 \leq x \leq 50$	+20
	CE ₃ :	$51 \leq x \leq 85$	+80
	CE ₄ :	$86 \leq x \leq 100$	+90
Valor de porcentaje INVÁLIDO	CE ₅ :	$x < 0$	-10
	CE ₆ :	$x > 100$	+200
	CE ₇ :	$x = \text{no entero}$	+1,5
	CE ₈ :	$x = \text{no numérico}$	A

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Tienda de Computadores /1

- **Análisis de la especificación**

- Parte del código de un programa trata el precio final de un artículo en base a su precio de venta al público, un descuento en % y el costo del envío (6,9 o 12 Euros, dependiendo del tipo de envío).

Variable	Clase de equivalencia	Estado	Representativo
Precio de venta al publico	CE _{1.1} : $x \geq 0$	Válido	1000,00
	CE _{1.2} : $x < 0$	Inválido	-1000,00
	CE _{1.3} : $x = \text{no numérico}$	Inválido	A
Descuento	CE _{2.1} : $0\% \leq x \leq 100\%$	Válido	10%
	CE _{2.2} : $x < 0\%$	Inválido	-10%
	CE _{2.3} : $x > 100\%$	Inválido	200%
	CE _{2.4} : $x = \text{no numérico}$	Inválido	A
Precio del envío	CE _{3.1} : $x = 6$	Válido	6
	CE _{3.2} : $x = 9$	Válido	9
	CE _{3.3} : $x = 12$	Válido	12
	CE _{3.4} : $x \neq \{6,9,12\}$	Inválido	4
	CE _{3.5} : $x = \text{no numérico}$	Inválido	A

Suposiciones

- El precio de venta de un artículo esta dado por un número con dos decimales
- El descuento es el valor porcentual sin decimales entre 0% y 100%
- El precio del envío puede ser 6, 9 o 12

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Tienda de Computadores /2

- **Casos de prueba para CE válidas**
 - Las clases de equivalencia válidas aportan las siguientes combinaciones o casos de prueba: T01. T02 y T03.

Variable	Clase de equivalencia	Estado	Representativo	T01	T02	T03
Precio de venta al publico	CE _{1.1} : $x \geq 0$	Válido	1000,00	x	x	x
	CE _{1.2} : $x < 0$	Inválido	-1000,00			
	CE _{1.3} : $x = \text{no numérico}$	Inválido	A			
Descuento	CE _{2.1} : $0\% \leq x \leq 100\%$	Válido	10%	x	x	x
	CE _{2.2} : $x < 0\%$	Inválido	-10%			
	CE _{2.3} : $x > 100\%$	Inválido	200%			
	CE _{2.4} : $x = \text{no numérico}$	Inválido	A			
Precio del envío	CE _{3.1} : $x = 6$	Válido	6	x		
	CE _{3.2} : $x = 9$	Válido	9		x	
	CE _{3.3} : $x = 12$	Válido	12			x
	CE _{3.4} : $x \neq \{6,9,12\}$	Inválido	4			
	CE _{3.5} : $x = \text{no numérico}$	Inválido	A			

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Tienda de Computadores /2

- **Casos de prueba para CE inválidas**

- Los siguientes casos de prueba han sido generados utilizando CE no válidas, cada una en combinación con CE válidas de otros elementos:

Variable	Clase de equivalencia	Estado	Representativo	T04	T05	T06	T07	T08	T09	T10
Precio de venta al publico	CE _{1.1} : $x \geq 0$	Válido	1000,00			x	x	x	x	x
	CE _{1.2} : $x < 0$	Inválido	-1000,00	x						
	CE _{1.3} : $x = \text{no numérico}$	Inválido	A		x					
Descuento	CE _{2.1} : $0\% \leq x \leq 100\%$	Válido	10%	x	x				x	x
	CE _{2.2} : $x < 0\%$	Inválido	-10%			x				
	CE _{2.3} : $x > 100\%$	Inválido	200%				x			
	CE _{2.4} : $x = \text{no numérico}$	Inválido	A					x		
Precio del envío	CE _{3.1} : $x = 6$	Válido	6	x	x	x	x	x		
	CE _{3.2} : $x = 9$	Válido	9							
	CE _{3.3} : $x = 12$	Válido	12							
	CE _{3.4} : $x \neq \{6,9,12\}$	Inválido	4						x	
	CE _{3.5} : $x = \text{no numérico}$	Inválido	A							x

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Ejemplo Tienda de Computadores /2

- **Todos los casos de pruebas**

- Se obtienen 10 casos de prueba: 3 casos de prueba positivos (valores válidos) y 7 casos de prueba negativos (valores no válidos).

Variable	Clase de equivalencia	Estado	Representativo	T01	T02	T03	T04	T05	T06	T07	T08	T09	T10
Precio de venta al publico	CE _{1.1} : $x \geq 0$	Válido	1000,00	x	x	x			x	x	x	x	x
	CE _{1.2} : $x < 0$	Inválido	-1000,00				x						
	CE _{1.3} : $x = \text{no numérico}$	Inválido	A					x					
Descuento	CE _{2.1} : $0\% \leq x \leq 100\%$	Válido	10%	x	x	x	x	x				x	x
	CE _{2.2} : $x < 0\%$	Inválido	-10%						x				
	CE _{2.3} : $x > 100\%$	Inválido	200%							x			
	CE _{2.4} : $x = \text{no numérico}$	Inválido	A								x		
Precio del envío	CE _{3.1} : $x = 6$	Válido	6	x			x	x	x	x	x		
	CE _{3.2} : $x = 9$	Válido	9		x								
	CE _{3.3} : $x = 12$	Válido	12			x							
	CE _{3.4} : $x \neq \{6,9,12\}$	Inválido	4									x	
	CE _{3.5} : $x = \text{no numérico}$	Inválido	A										x

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Basada en la salida (output based)

- Las clases de equivalencia también pueden ser generadas a partir de los valores de salida esperados.
 - El método utilizado es análogo al anterior, aplicando los valores de salida.
 - La variable (elemento) es entonces la salida (output), por ejemplo, el valor de un campo en la Interfaz Gráfica de usuario GUI").
 - Las clases de equivalencia son generadas para todos los posibles valores de la salida definidos.
 - Se determina un representante para cada clase de equivalencia de los valores de salida.
- Entonces, el valor de entrada, que conduce al valor representante es obtenido/identificado. Mayores costos y esfuerzo dado que los valores de entrada deben ser obtenidos para una salida determinada de forma recursiva.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – En general /1

- **Partición.**

- La calidad de la prueba depende de la segmentación precisa de variables/elementos en clases de equivalencia.
- Las CE que no hubieran sido Identificadas presentan el riesgo de posibles omisiones, dado que los representantes utilizados no cubren todas las posibilidades.

- **Casos de prueba.**

- El método de la clase de equivalencia aporta casos de prueba para los cuales aun debe ser seleccionado un representante.
- Las combinaciones de datos de prueba son seleccionadas definiendo el o los representantes de cada clase de equivalencia.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – En general /2

- Selección de representantes.
 - Cada valor perteneciente a la CE puede ser un representante. Los óptimos son:
 - Valores característicos (valores típicos o utilizados con frecuencia).
 - Valores problemáticos (valores que se sospecha pueden producir fallos)
 - Valores límite (valores en la frontera de la CE).
 - Durante la creación de casos de prueba, los representantes de cada CE inválidos deben combinarse siempre con los mismos valores de otros CE válidos (combinaciones estándar).
 - Dado el posible enmascaramiento de errores, se debe evitar las combinaciones de representantes de CE inválidas con representantes de otras CE inválidas en un mismo caso de prueba.
 - La selección de representantes implica que la función implementada por el programa/sistema utiliza comparadores.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Cobertura

- La cobertura de clases de equivalencia puede ser utilizada como criterio de salida para finalizar las actividades del proceso de pruebas.

$$Cobertura (CE) = \frac{\text{número de CE probados}}{\text{número de CE definidos}} * 100 \%$$

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Transición

- La transición de la especificación o definición de la funcionalidad para la creación de la clase de equivalencia.
 - Con frecuencia es una tarea difícil debido a la carencia de documentación precisa y completa.
 - Los límites no definidos o las descripciones faltantes hacen difícil la definición de las clases de equivalencia.
 - Con frecuencia, es necesario mantener contacto con el cliente con el objeto de completar la información.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Partición de equivalencia – Beneficios

- Método sistemático para el diseño de casos de prueba, por ejemplo, con una mínima cantidad de casos de prueba se puede esperar un valor de cobertura específico.
- La partición del rango de valores en clases de equivalencia a partir de las especificaciones cubre los requisitos funcionales.
- La asignación de prioridades a la clases de equivalencia puede ser utilizada para la asignación de prioridades a los casos de prueba (los valores de entrada utilizados con poca frecuencia deben ser los últimos en ser probados).
- Las pruebas de las excepciones conocidas está cubierta por los casos de prueba de las clases de equivalencia negativas.
- La partición de equivalencias es aplicable en todos los niveles de pruebas.
- La partición de equivalencias puede ser usada para lograr la cobertura de los objetivos de entrada y salida.
- Puede ser aplicado por entradas manuales usando las interfaces gráficas de usuario (GUI) o por las interfaces de parámetros en las pruebas de integración.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box)

- Partición de equivalencias o clases de equivalencia.
- **Análisis de valores límite.**
- Tablas de decisión y gráficos causa y efecto.
- Pruebas de transición de estado.
- Pruebas de caso de uso.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite – Generalidades /1

- El análisis de valores límite amplía la técnica de partición en clases de equivalencia introduciendo una regla para seleccionar a los representantes.
- Los valores frontera (valores límite) de la clase de equivalencia deben ser probados de forma intensiva.
- **¿Porqué prestar más atención a los límites?**
 - Frecuentemente los límites del rango de valores no están bien definidos o conducen a distintas interpretaciones.
 - Comprobar si los límites han sido implementados (programados) correctamente
- **Importante:**
 - La experiencia demuestra que con mucha frecuencia, los errores tienen lugar en los límites del rango de valores.

El análisis de valores límite puede ser aplicado en todos los niveles de prueba. Es fácil de aplicar y tiene una alta capacidad de encontrar defectos basándose en el detalle de las especificaciones.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite – Generalidades /1

- El análisis de valores límite supone que:
 - La clase de equivalencia está compuesta de un rango continuo de valores (no por un valor Individual o un conjunto de valores discretos).
 - Se pueden definir los límites para el rango de valores.
- Por ser una extensión a la partición en clases de equivalencia el análisis de valores límite es un método que sugiere la selección de representantes.
 - Partición en clases de equivalencia:
 - Evalúa un valor típico de lo clase de equivalencia
 - Análisis de valores límite:
 - Evalúo los valores límite (frontera) y su entorno.
 - Se utiliza el siguiente esquema:

Valor del rango: Valor Mínimo $\leq x \leq$ Valor Máximo		
Valor mínimo - δ	límite inferior	Valor mínimo + δ
Valor máximo - δ	límite superior	Valor máximo + δ
δ = menor incremento definido para el valor		
Por ejemplo: 1 para valores enteros		

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite - Definición de valores límite

- El esquema básico sólo puede ser aplicado cuando el rango de valores ha sido definido de conformidad con el mismo esquema.
 - En este caso no son necesarias pruebas adicionales para un valor intermedio del rango de valores.
- Si un CE está definido como un único valor numérico, por ejemplo $x = 5$, los valores correspondientes al entorno también serán utilizados.
 - Los representantes (de la clase y su entorno) son: 4 - 5 - 6.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite - CE no válidas

- Los valores límite para clases de equivalencia no válidas tienen poco sentido.
 - Los representantes de una CE no válido en la frontera de una CE válida ya se encuentran cubiertas a través del esquema básico.
- Para rangos de valores definidos como un conjunto de valores, en general, no es posible definir los valores límite.
 - Por ejemplo: Soltero, casado, divorciado, viudo.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite – Ejemplo rango de descuento 1

- El rango de valores para un descuento está definido entre: $0,00 \% \leq x \leq 100,00\%$
- Definición de CE
3 clases:
 - CE: $x < 0,99$
 - CE: $0,00 \leq x \leq 100,00$
 - CE: $x > 100,01$
- Análisis de valores límite.
Extiende los representantes a:
 - CE: | -0,99 | 0,00 | 0,01 | 99,99 | 100,00 | 100,01 |
- Nota importante:
En lugar de un representante para la CE válida, ahora hay seis representantes (cuatro válidos y dos no válidos).

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Análisis de valores límite – Ejemplo rango de descuento 2

- Esquema básico: seleccionar tres valores con el objeto de ser probados - el valor límite exacto y dos valores pertenecientes al entorno (dentro y fuera de la CE).
- Punto de vista alternativo: dado que el valor límite pertenece a la CE, solo son necesarios dos valores para las pruebas, uno perteneciente a la CE y otro no perteneciente a la CE.
- **El rango de valores para un descuento está definido entre: $0,00 \% \leq x \leq 100,00\%$**
 - CE válida: $0,00 \leq x \leq 100,00$
 - Análisis de valores límite:
 - Los representantes adicionales son:
| -0,99 | 0,00 | 0,01 | 99,99 | 100,00 | 100,01 |
 - 0,01 tiene el mismo comportamiento que 0,00
 - 99,99 tiene el mismo comportamiento que 100,00
- Un error de programación causado por un operador de comparación erróneo será detectado con los dos valores límite (frontera).

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box)

- Partición de equivalencias o clases de equivalencia.
- Análisis de valores límite.
- **Tablas de decisión y gráficos causa y efecto.**
- Pruebas de transición de estado.
- Pruebas de caso de uso.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Definición /1

- La partición en clases de equivalencia y el análisis de valores límite tratan entradas en condiciones aisladas.
- Sin embargo, una condición de entrada puede tener efectos sólo en combinación con otras condiciones de entrada.
- Todos los métodos descritos previamente no tienen en cuenta el efecto de dependencias y combinaciones.
- El uso del conjunto completo de las combinaciones de todas las clases de equivalencia de entrada, conduce normalmente a un número muy alto de casos de prueba (explosión de casos de prueba).
- Con la ayuda de diagramas causa y efecto y tablas de decisiones, se puede determinar la cantidad de combinaciones posibles y se pueden reducir de forma sistemática a un subconjunto.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Definición /2

El diagrama causa y efecto utiliza un lenguaje formal.


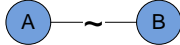
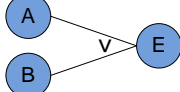
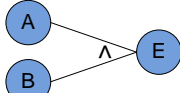
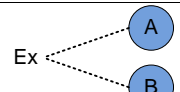
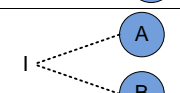
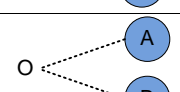
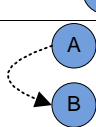
El diagrama causa y efecto se genera traduciendo la especificación (normalmente informal) de un objeto de prueba a un lenguaje formal.

El objeto de prueba está sometido a una determinada cantidad de efectos que se remontan a sus respectivas causas.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Símbolos

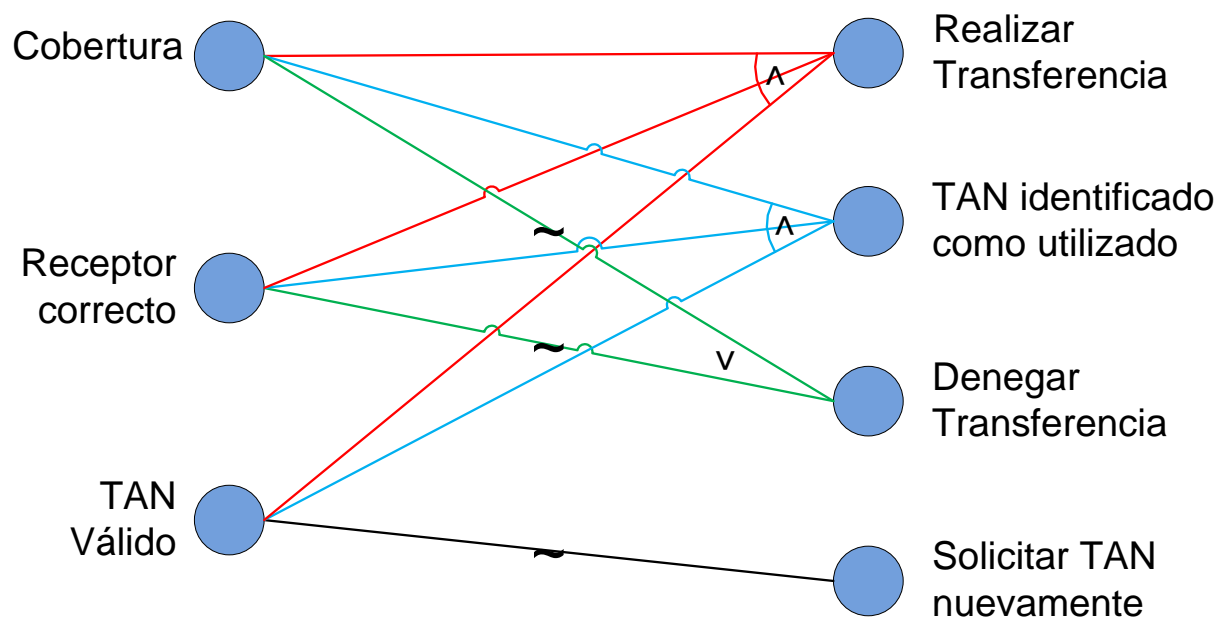
Nombre	Definición	Símbolo
aseveración (assertion)	Si causa A, entonces efecto E.	
negación (negation)	Si causa A, entonces no efecto E	
o (or)	Si causa A o B, entonces efecto E	
y (and)	Si causa A y B, entonces efecto E	
exclusivo (exclusive)	o causa A, o causa B	
inclusivo (inclusive)	Por lo menos una de las 2 causas, A o B	
uno y solo uno (one and only one)	Una y exactamente una de las 2 causas, A o B	
requerido (required)	Si causa A entonces causa B	

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Ejemplo Banca Online /1

El usuario se Identifica a través de su número de cuenta y PIN. Si tiene suficiente cobertura podrá realizar una transferencia. Para poder realizar la transferencia debe introducir los datos del receptor y un TAN (Número de Autenticación de Transferencia) válido.



IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Ejemplo Banca Online /2

Tabla de decisión	Descripción	T01 T02	T03	T04	T05
Causa	Suficiente cobertura	Sí	No	-	-
	Receptor correcto	Sí	-	No	-
	TAN válido	Sí	-	-	No
Efecto	Realizar Transferencia	Sí	No	No	No
	Marcar TAN como utilizado	Sí	No	No	No
	Denegar transferencia	No	Sí	Sí	No
	Solicitar TAN nuevamente	No	No	No	Sí

- Cada columna de la tabla representa un caso de prueba.
- Construcción de la tabla de decisión:
 - Seleccionar un efecto.
 - Regresar al diagrama para identificar las causas.
 - Cada combinación de causas está representada por una columna en la tabla de decisión (un caso de prueba).
 - Combinaciones de causas idénticas que conducen a efectos distintos, se pueden fusionar para formar un único caso de prueba.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Uso Práctico

- La especificación está dividida en partes fáciles de gestionar, por lo que conlleva a una tabla de decisión con un tamaño práctico.
- Es difícil deducir valores límite a partir del diagrama causa y efecto o de la tabla de decisión.
- Es recomendable combinar casos de prueba obtenidos a partir de tablas de decisión con los obtenidos a partir de un análisis de valores límite.
- El número de causas y efectos analizados determinarán la complejidad del diagrama causa y efecto: para n precondiciones cuyos posibles valores puedan ser verdadero o falso, se pueden generar 2^n casos de prueba.
- Para sistemas de mayor tamaño este método sólo es controlable con el apoyo de herramientas.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Tablas de decisión y gráficos causa y efecto – Beneficios y Desventajas

- **Beneficios.**

- Identificación sistemática de combinaciones de entradas (combinaciones de causas) que no podrían ser identificados utilizando otros métodos.
- Los casos de prueba son fáciles de obtener a partir de la tabla de decisión.
- Facilidad de determinar una cobertura suficiente de casos de prueba, por ejemplo, por lo menos un caso de prueba por cada columna de la tabla de decisión.
- El número de casos de prueba se puede reducir por la fusión sistemática de columnas de la tabla de decisión.

- **Desventajas.**

- El establecimiento de un gran número de causas conduce a resultados complejos y extensos.
- Por lo tanto, se puede incurrir en muchos errores en la aplicación de este método.
- Esto hace necesario el uso de una herramienta.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box)

- Partición de equivalencias o clases de equivalencia.
- Análisis de valores límite.
- Tablas de decisión y gráficos causa y efecto.
- **Pruebas de transición de estado.**
- Pruebas de caso de uso.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de transición de estado – Definición

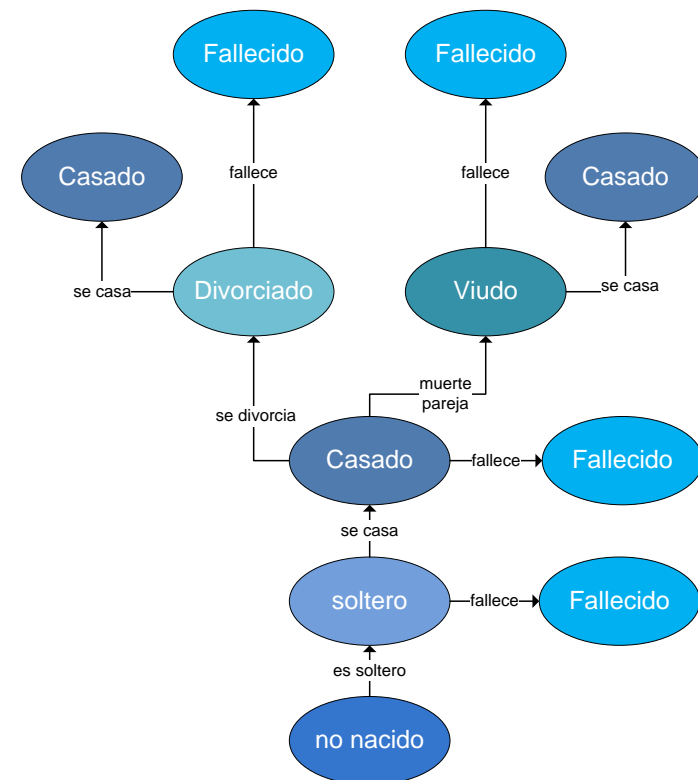
- Muchos métodos sólo tienen en cuenta el comportamiento del sistema en términos de datos de entrada (input data) y datos de salida (output data).
- No se tiene en cuenta los diferentes estados que pueda tomar el objeto de prueba.
 - Por ejemplo, el resultado de acciones que hubieran ocurrido en el pasado, acciones que hubieran causado que el objeto de prueba adquiriera un determinado estado Interno.
- Los distintos estados que puede tomar un objeto de prueba se modelan a través de diagramas de transición de estado.
- El análisis de la transición de estado se utiliza para definir casos de prueba basados en la transición de estado.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de transición de estado – Árbol de Transición de Estados

- Para determinar casos de prueba usando un gráfico de transición de estado, se construye un árbol de estados:
 - La raíz del árbol es el estado inicial
 - Para todos los estados, las posibles transiciones de un estado a otro se representan como ramas.
 - Para cada estado que puede ser alcanzado desde el estado inicial, se crea un nodo que está conectado a la raíz a través una rama.
 - Esta operación se repite para todos los estados sucesivos.
 - La creación de ramas finaliza si:
 - El estado correspondiente al nodo es un estado final (hoja del árbol)
 - Un mismo nodo con el mismo estado ya es parte del árbol.
- Las hojas del árbol o nodos finales representan un estado final.



IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de transición de estado – Tabla de Transición de Estados

- Cada camino o ruta desde la raíz hasta la hoja representa un caso de prueba para las pruebas de transición de estados
- Para el ejemplo del gráfico anterior tenemos 6 casos de prueba

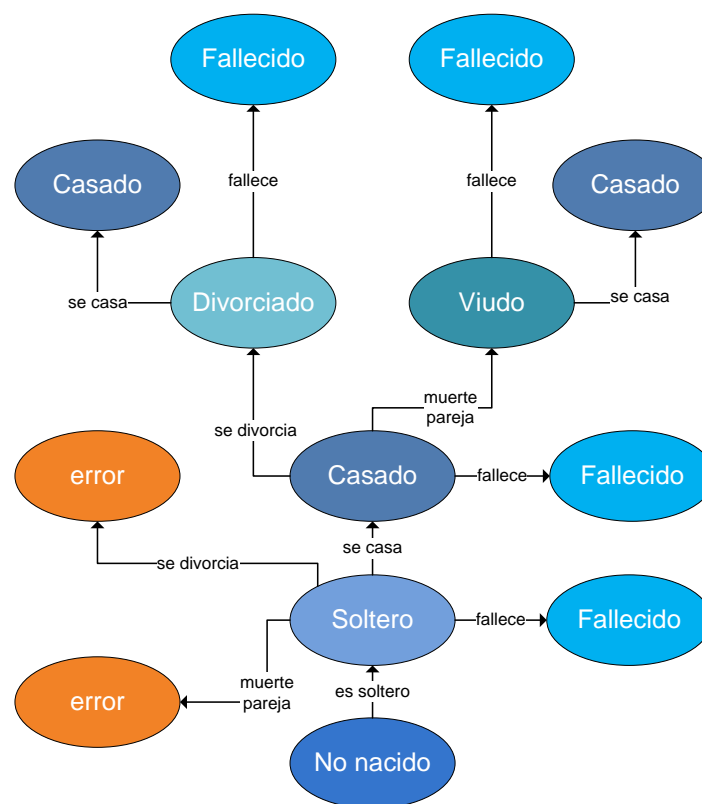
Casos de Prueba	Estado1 (Inicial)	Estado2	Estado3	Estado4	Estado5	Estado final
T01	no nacido	soltero	fallecido			fallecido
T02	no nacido	soltero	casado	fallecido		fallecido
T03	no nacido	soltero	casado	viudo	fallecido	fallecido
T04	no nacido	soltero	casado	viudo	casado	casado
T05	no nacido	soltero	casado	divorciado	fallecido	fallecido
T06	no nacido	soltero	casado	divorciado	casado	casado

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de transición de estado – Árbol de Transición de Estados

- El árbol de transición de estados de nuestro ejemplo puede ahora ser extendido usando transiciones inválidas (casos de prueba negativos, pruebas de robustez).
- Se muestran dos posibles transiciones inválidas (entre otras)
- Las transiciones imposibles entre estados no pueden ser probadas



IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de transición de estado – Resumen

- Criterio de salida de la prueba.
 - Cada estado debe haber sido alcanzado al menos una vez.
 - Cada transición debe haber sido ejecutada al menos una vez.
- Beneficios / Desventajas de este método.
 - Buen método de pruebas para aquellos objetos de prueba que pueden ser descritos como una máquina de estados.
 - Buen método de pruebas para probar clases, sólo en el caso de disponer del ciclo de vida del objeto.
 - Con frecuencia, los estados son más bien complejos, es decir, es necesario un gran número de parámetros para describir el estado.
 - En estos casos, diseñar casos de prueba y analizar los resultados de las pruebas puede ser difícil y puede implicar un consumo de tiempo considerable.
 - La sola cobertura de todos los estados no garantiza una cobertura completa de las pruebas.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box)

- Partición de equivalencias o clases de equivalencia.
- Análisis de valores límite.
- Tablas de decisión y gráficos causa y efecto.
- Pruebas de transición de estado.
- **Pruebas de caso de uso.**

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de caso de uso – Definición

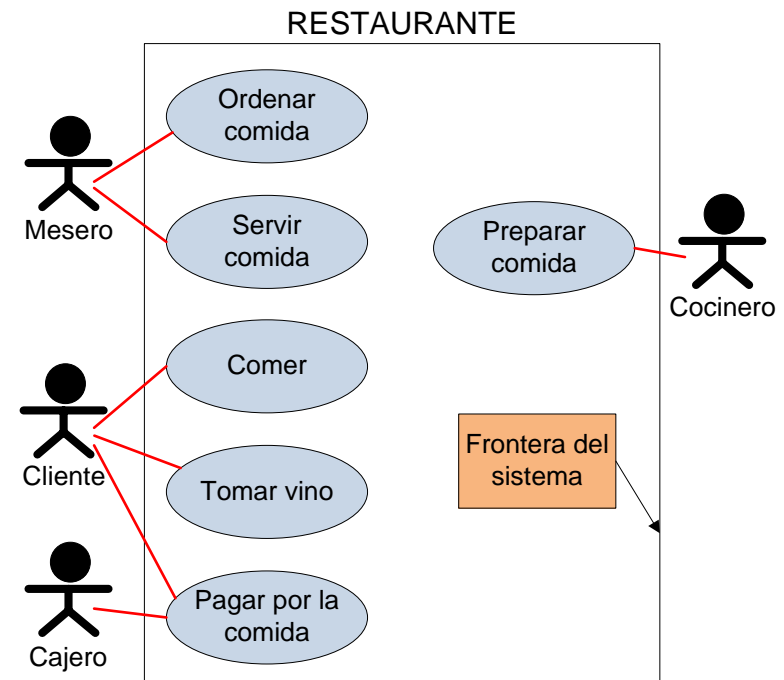
- Los casos de prueba se obtienen directamente a partir de los casos de uso del objeto de prueba.
 - El objeto de prueba es visto como un sistema reaccionando con sus actores.
 - Un caso de uso describe la interacción de todos los actores involucrados conduciendo a un resultado final por parte del sistema.
 - Todo caso de uso tiene **precondiciones** que deben ser cumplidas con el objeto de ejecutar el caso de uso (caso de prueba) de forma satisfactoria.
 - Todo caso de uso tiene **poscondiciones** que describen el sistema tras la ejecución del caso de uso (caso de prueba).
- Los casos de uso son elementos del Lenguaje Unificado de Modelado (Unified Modeling Language - UML).
 - El diagrama de casos de uso es uno de los 13 diferentes tipos de diagramas utilizados por UML.
 - Un diagrama de casos de uso describe un comportamiento, no describe una secuencia de eventos.
 - Un diagrama de casos de uso muestra la reacción del sistema desde el **punto de vista del usuario**.
- **UML es un lenguaje de especificación propietario para el modelado de objetos**

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de caso de uso – Ejemplo de Restaurante (fuente: Wikipedia).

- El diagrama describe la funcionalidad de un Sistema de Restaurante sencillo.
- Los casos de uso están representados por óvalos y los actores están representados por figuras humanas.
- El actor **Cliente** puede comer comida, pagar por la comida o beber Vino.
- El actor **Cocinero** sólo puede preparar comida.
- Los actores **Cliente** y **Cajero** están involucrados en el caso de uso pagar por la comida.
- La caja define los límites del sistema del Restaurante, es decir, los casos de uso representados son parte del sistema a modelar y no los actores.



IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de caso de uso – Generalidades

- Cada caso de uso describe una cierta tarea (interacción usuario - sistema).
- La descripción de un caso de uso incluye, pero no está limitado a:
 - Precondiciones.
 - Resultados esperados / comportamiento del sistema.
 - Poscondiciones.
- Estos elementos descriptivos también son utilizados para definir el caso de prueba correspondiente.
- Cada caso de uso puede ser utilizado como la fuente para un caso de prueba.
- Cada alternativa en el diagrama corresponde a un caso de prueba separado.
- Normalmente la Información aportada por un caso de uso no tiene suficiente detalle para definir casos de prueba. Son necesarios datos adicionales (datos de entrada, resultados esperados) para construir o desarrollar un caso de prueba.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Pruebas de caso de uso – Resumen

- **Beneficios.**

- Pruebas apropiadas Para pruebas de aceptación y pruebas de sistema, dado que cada caso describe un escenario a probar.
- Pruebas apropiadas si las especificaciones del sistema se encuentran disponibles en UML.

- **Desventajas.**

- Nula obtención de casos de prueba adicionales más allá de la información aportada por el caso de uso.
- Por lo tanto este método debería ser utilizado sólo en combinación con otros métodos de diseño sistemático de casos de prueba.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box) o basadas en la especificación – Conclusiones Generales /1

- El objetivo principal de las pruebas de caja negra (black box) es probar la funcionalidad del sistema.
 - Por lo tanto, el resultado de las pruebas depende de la calidad de la especificación del sistema (por ejemplo, la completitud, especificaciones faltantes o erróneas conducen a malos casos de prueba).
 - Si las especificaciones son erróneas, también serán erróneos los casos de prueba. Las pruebas se desarrollan solamente para las funciones descritas. Una especificación faltante de una funcionalidad requerida no será detectada durante las pruebas.
- Si el objeto de prueba posee funciones que no han sido especificadas, éstas no serán evaluadas.
 - Tales funciones superfluas pueden causar problemas en las áreas de la estabilidad y seguridad (por ejemplo, software para un cajero automático).

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box) o basadas en la especificación – Conclusiones Generales /2

- A pesar de estas desventajas, las pruebas funcionales constituyen la actividad de pruebas más importante.
 - Los métodos de caja negra (black box) siempre son utilizados en el proceso de pruebas.
 - Las desventajas pueden ser compensadas con el uso de métodos adicionales de diseño de casos de prueba, por ejemplo, pruebas de caja blanca o pruebas basadas en la experiencia.

IV – Técnicas de Diseño de Pruebas

03 – Técnicas de caja negra (black box) o basadas en la especificación

Técnicas de caja negra (black box) o basadas en la especificación – Resumen

- Métodos de caja negra (black box):
 - Partición de equivalencias o clases de equivalencia.
 - Análisis de valores límite.
 - Tablas de decisión y gráficos causa y efecto.
 - Pruebas de transición de estado.
 - Pruebas de caso de uso.
- Las pruebas de caja negra (black box) verifican funciones especificadas, si los funciones no son especificadas, éstas no son probadas.
- El código sobrante no puede ser detectado utilizando pruebas de caja negra (black box).

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

- IV/01 Pruebas en el proceso de desarrollo
- IV/02 Categorías de las técnicas de diseño de pruebas.
- IV/03 Técnicas de caja negra (black box) o basadas en la especificación.
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- IV/05 Técnicas basadas en la experiencia.
- IV/06 Selección de las técnicas de pruebas.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Técnicas de caja blanca (white box) o basadas en la estructura – Visión General /1

- En el presente apartado se explicarán en detalle las siguientes técnicas de caja blanca (white box):
 - Pruebas de sentencia y cobertura.
 - Pruebas de decisión o rama (branch) y cobertura.
 - Pruebas de camino (path) y cobertura.
 - Pruebas de condición y cobertura.
- **Observación:**
 - Estas técnicas representan las técnicas de pruebas dinámicas más importantes y utilizadas de forma más frecuente. Estas técnicas están relacionadas con las técnicas de análisis estático descritas anteriormente.
- Otras técnicas de caja blanca (pero no limitadas a estas) son:
 - LCSAJ (linear Code Sequence And Jump): Secuencia lineal de código y salto.
 - Técnicas basadas en el flujo de datos.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Técnicas de caja blanca (white box) o basadas en la estructura – Visión General /2

- Se basa en identificar la estructura del software o el sistema:
 - Nivel de Componente: La estructura de un componente de software, por ejemplo: sentencias, decisiones, ramas, caminos (paths).
 - Nivel de Integración: La estructura puede ser llamada árbol (un diagrama con módulos y llamados a otros módulos).
 - Nivel de Sistema: La estructura puede ser un menú, procesos de negocio, o la estructura de una página web.
- Las tres anteriores están relacionadas con las técnicas diseño de las pruebas de estructura por la cobertura del código basada en sentencias, decisiones y caminos.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Técnicas de caja blanca (white box) o basadas en la estructura – Herramientas /1

- Durante los pruebas de caja blanca, el programa objeto de pruebas es ejecutado de misma forma que las pruebas de caja negra. Ambas categorías (caja blanca y caja negra) conforman las pruebas dinámicas.
 - La teoría establece que todas las partes de un programa deberían ser ejecutadas por lo menos una vez.
- El grado de cobertura de un programa se mide con el uso de herramientas (por ejemplo, analizadores de cobertura):
 - La instrumentación del código se lleva a cabo con el objeto de contar la ejecución de caminos, es decir se insertan contadores en el código del programa del objeto de prueba.
 - Estos contadores son inicializados en cero, cada ejecución del camino específico Incrementará el contador correspondiente.
 - Los contadores que mantienen el valor cero tras las pruebas indican las partes del programa que aún no han sido ejecutadas.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Técnicas de caja blanca (white box) o basadas en la estructura – Herramientas /2

- Las técnicas de caja blanca requieren el apoyo de herramientas en muchas áreas a saber:
 - Especificación de caso de prueba.
 - Generación automática del diagrama del flujo de control a partir del código fuente del programa.
 - Ejecución de la prueba.
 - Herramientas para monitorizar y controlar el flujo del programa dentro del objeto de prueba.
- El soporte de herramientas asegura la calidad de las pruebas e incrementa su eficiencia.
 - Dada la complejidad de las mediciones necesarias para las pruebas de caja blanca, la ejecución manual de pruebas implica:
 - Consumo excesivo de tiempo y de recursos.
 - Dificultad en la implementación y propensión o errores.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Pruebas de caja blanca (White box)

- **Cobertura de sentencia.**
- Cobertura de decisión o rama (branch).
- Cobertura de camino (path).
- Cobertura de condición.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia (Statement coverage) - Definición

- El foco de la atención son las sentencias del código de un programa.
 - ¿Qué casos de prueba son necesarios con el objeto de ejecutar todas (o un porcentaje determinado) las sentencias del código existentes?
- La base de este análisis es el diagrama de flujo de control.
 - Todas las instrucciones o sentencias están representadas por nodos y el flujo de control entre instrucciones está representado por una arista (flecha).
 - Las instrucciones múltiples se combinan en un nodo independiente si solamente pueden ser ejecutados en una secuencia particular.
- El objetivo de la prueba (criterio de salida) es lograr la cobertura de un porcentaje específico de todas las sentencias denominado cobertura de sentencia. (C_0 cobertura de código - code coverage).

$$\text{Cobertura de Sentencia } (C_0) = \frac{\text{número de sentencias ejecutadas}}{\text{número total de sentencias}} * 100\%$$

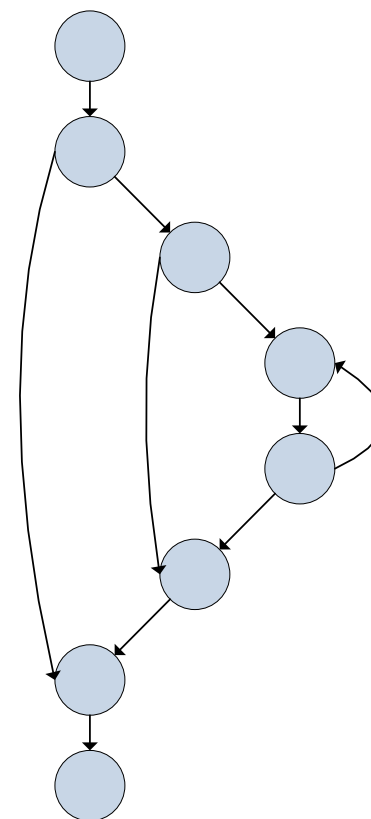
IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia (Statement coverage) – Ejemplo (1) /1

- Se evalúa el siguiente segmento de código de un programa, que está representado por el diagrama del flujo de control.

```
if (i > 0) {  
    j = f(i);  
    if (j > 10){  
        while (k > 10){  
            ...  
        }  
    }  
}
```

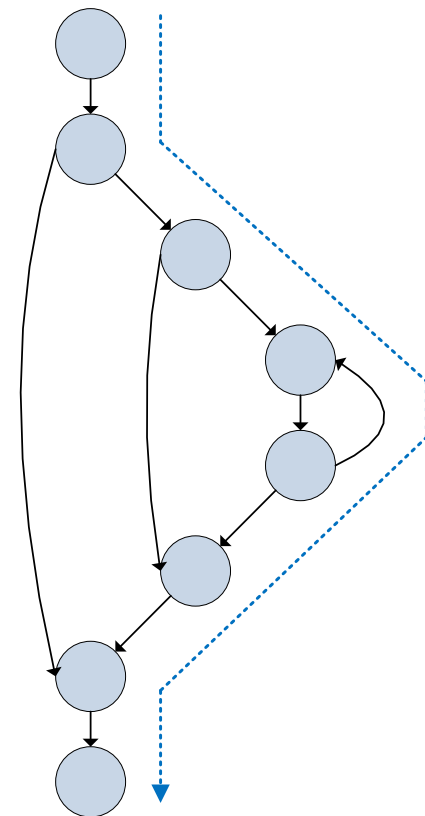


IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia (Statement coverage) – Ejemplo (1) /2

- Considerar el programa representado por el diagrama de flujo de control.
 - Contiene dos sentencias **if** y un bucle **do-while** dentro de la segunda sentencia **if**.
- Hay tres caminos diferentes a través del segmento de programa.
 - La primera sentencia **if** permite dos direcciones.
 - La dirección de la derecha de la primera sentencia **if** se divide nuevamente a partir de una segunda sentencia **if**.
- Todas las sentencias de este programa pueden ser alcanzadas haciendo uso del camino de la derecha.
 - Un solo caso de prueba será suficiente para alcanzar el 100% de cobertura de sentencia.

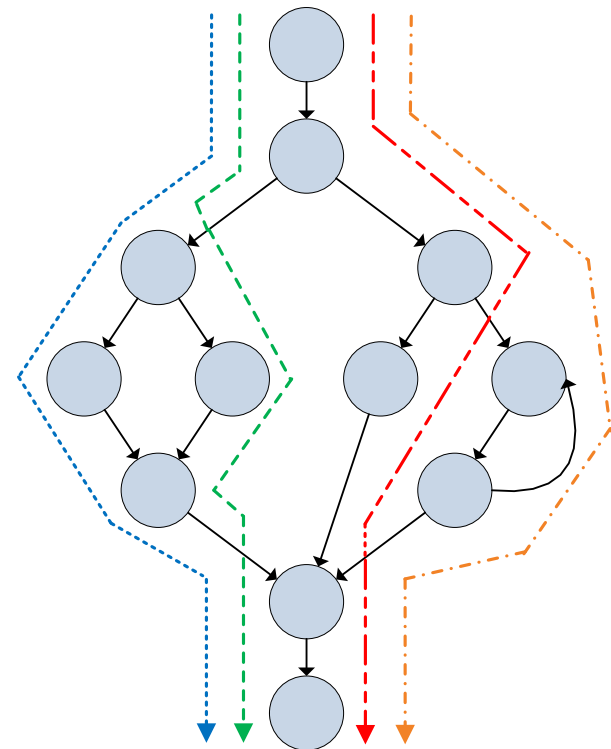


IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia (Statement coverage) – Ejemplo (2)

- En este ejemplo el diagrama es ligeramente más complejo:
 - El programa contiene las sentencias **if** y un bucle (dentro de una sentencia **if**)
- Cuatro rutas diferentes conducen a la cobertura de sentencias de este segmento de programa.
 - La primera sentencia **if** permite dos direcciones.
 - En cada rama de la sentencia **if**, existe otra sentencia **if** que permite dos direcciones diferentes.
- Utilizando un caso de prueba (azul, verde o naranja) un máximo de 7 de 12 sentencias pueden ser cubiertas, esto resulta en un valor de $C_0 = 58,33\%$
- Utilizando el caso de prueba (rojo) un máximo de 6 de 12 sentencias pueden ser cubiertas, esto resulta en un valor de $C_0 = 50\%$
- Para una cobertura de sentencia del 100% hacen falta cuatro casos de prueba.



IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia (Statement coverage) – Conclusiones generales

- La medición de la cobertura se realiza con el uso de herramientas diseñadas de forma específica.
 - Estas herramientas se denominan Herramientas de Análisis de Cobertura (Coverage Analysis Tools) o Analizadores de Cobertura (Coverage Analyzers).
- **Beneficios/desventajas de este método.**
 - El código muerto (código constituido por sentencias que nunca se ejecutan) será detectado.
 - Si hay código muerto en el programa, no se podrá lograr una cobertura del 100%.
 - Instrucciones faltantes (código faltante que es necesario para cumplir con la especificación) no puede ser detectado.
 - Las pruebas se desarrollan solamente para las sentencias ejecutadas: ¿Todo el código puede ser alcanzado/ejecutado?
 - El código faltante no puede ser detectado utilizando técnicos de caja blanca.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Pruebas de caja blanca (White box)

- Cobertura de sentencia.
- **Cobertura de decisión o rama (branch).**
- Cobertura de camino (path).
- Cobertura de condición.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de decisión o rama (decision coverage) – Definición

- En lugar de las sentencias, la cobertura de decisión se centra en el flujo de control en un segmento de programa (no los nodos sino las aristas del diagrama de flujo de control).
 - Todas las aristas del diagrama de flujo de control tienen que ser cubiertas al menos una vez.
 - ¿Qué casos de prueba son necesarios para cubrir cada arista del diagrama de flujo de control al menos una vez?
- El propósito de esta prueba (criterio de salida) es lograr la cobertura de un porcentaje específico de todas las decisiones denominado cobertura de decisión (C_1 cobertura de código - code coverage)

$$\text{Cobertura de Decisión } (C_1) = \frac{\text{número de decisiones ejecutadas}}{\text{número total de decisiones}} * 100\%$$

Sinónimo de:

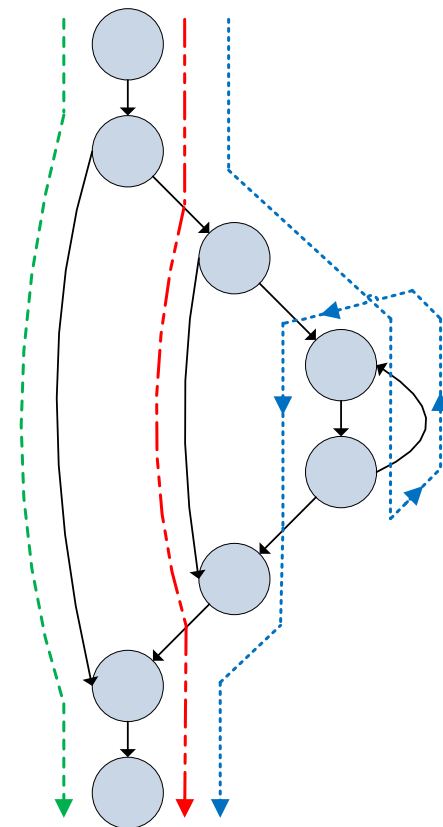
$$\text{Cobertura de Ramas } (C_1) = \frac{\text{número de ramas cubiertas}}{\text{número total de ramas}} * 100\%$$

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de decisión o rama (decision coverage) – Ejemplo (1)

- El diagrama de flujo de control representa el segmento de un programa objeto de la evaluación.
- Tres caminos diferentes son posibles en este segmento de programa.
 - La primera sentencia **if** conduce a dos direcciones diferentes.
 - Un camino de la primera sentencia **if** se divide nuevamente en dos caminos diferentes, uno de los cuales contiene un bucle.
 - Solamente se puede alcanzar todas las aristas a través de una combinación de los tres caminos posibles.
 - Son necesarios tres casos de prueba para alcanzar una cobertura de decisión del 100%.
 - Utilizando solamente las dos direcciones de la derecha pueden ser cubiertas 9 de 10 aristas ($C_1 = 90\%$)

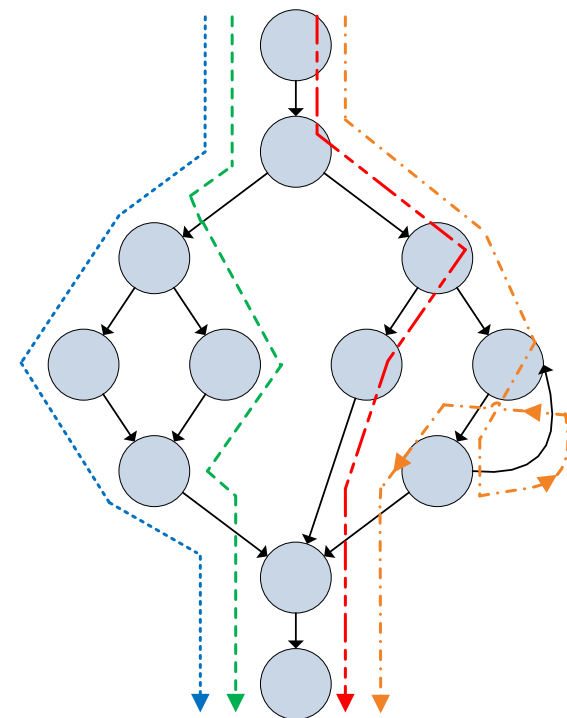


IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de decisión o rama (decision coverage) – Ejemplo (2)

- En este ejemplo el diagrama es ligeramente más complejo.
- Cuatro caminos diferentes conducen a través del segmento de programa.
 - La primera sentencia **if** permite dos direcciones.
 - En ambas ramas de la sentencia **if**, otra sentencia **if** permite nuevamente dos direcciones diferentes.
 - En este ejemplo, el bucle no se cuenta como una decisión adicional.
- Utilizando un caso de prueba (el naranja), pueden ser cubiertas 7 de 15 aristas. Esto resulta en un valor de $C_1=46,67\%$.
 - Son necesarios cuatro casos de prueba para lograr una cobertura de decisión del 100% (el mismo número de casos para lograr una cobertura de sentencia del 100%).



IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de decisión o rama (decision coverage) – Conclusiones Generales

- Lograr una cobertura de decisión del 100% requiere, al menos de los mismos casos de prueba que requiere la cobertura de sentencia (más en la mayoría de los casos).
 - Una cobertura de decisión del 100% siempre incluye una cobertura de sentencia del 100%.
- La mayoría de las aristas son cubiertas en múltiples ocasiones.
- **Desventajas.**
 - No se pueden detectar sentencias faltantes.
 - No es suficiente para probar condiciones complejas.
 - No es suficiente para probar bucles de forma extensiva.
 - No se consideran las dependencias entre bucles.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de sentencia y cobertura de decisión

- Ambos métodos se refieren a caminos a través del diagrama de flujo de control.
 - Difieren en la cantidad de casos de prueba necesarios para lograr el 100% de cobertura.
- Sólo se considera el resultado final de una condición, a pesar de que la condición resultante puede estar constituida por múltiples condiciones atómicas.
 - La condición **if ((a > 2) OR (b < 6))** sólo puede ser verdadera o falsa.
 - El camino (del programa) a ejecutar depende solamente del resultado final de la condición combinada.
 - Aquellos fallos debidos o una Implementación errónea de las partes de una decisión combinada pueden no ser detectados.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Pruebas de caja blanca (White box)

- Cobertura de sentencia.
- Cobertura de decisión o rama (branch).
- **Cobertura de camino (path).**
- Cobertura de condición.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de camino (path coverage) – Definición /1

- La cobertura de camino se centra en la ejecución de todos los posibles caminos a través de un programa.
 - Un camino es una combinación de segmentos de programa (en un diagrama de flujo de control es una secuencia alternada de nodos y aristas)
- Los bucles en la cobertura de caminos
 - Para la cobertura de decisión, un solo camino a través de un bucle es suficiente.
 - Para la cobertura de camino hay casos de prueba adicionales:
 - Un caso de prueba que no entre al bucle.
 - Un caso de prueba adicional por cada iteración en el bucle.
- Esto puede conducir o un alto número de casos de prueba.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de camino (path coverage) – Definición / 2

El foco del análisis de cobertura es el diagrama de flujo de control.

Las sentencias son nodos.

El flujo de control está representado por las aristas.

Cada camino es una vía única desde el inicio hasta el final del diagrama de flujo de control.

El objetivo de esta prueba (criterio de salida) es alcanzar un porcentaje definido de cobertura de caminos.

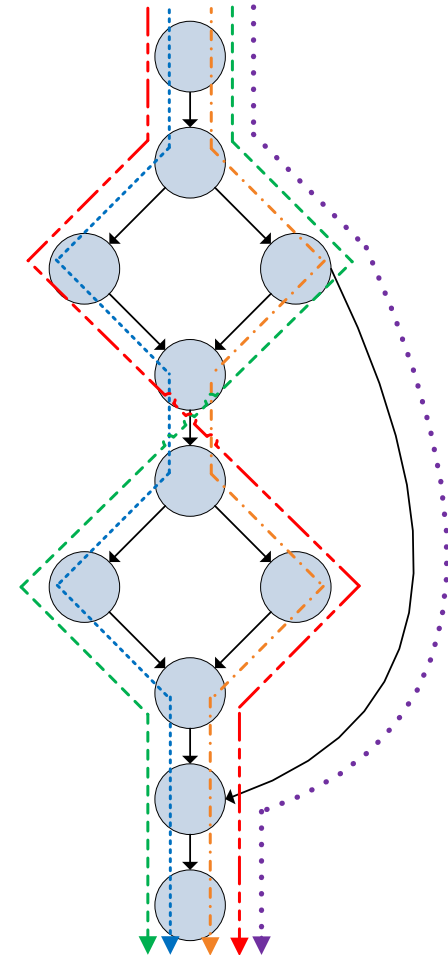
$$\text{Cobertura de Camino} = \frac{\text{número de caminos ejecutados}}{\text{número total de caminos}} * 100\%$$

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de camino (path coverage) – Ejemplo (1)

- El diagrama de flujo de control, representa el segmento de programa a ser evaluado. Contiene tres sentencias **if**.
- Tres caminos diferentes conducen a través del diagrama de este segmento de programa logran una cobertura de decisión completa.
- Sin embargo, pueden ser ejecutados cinco posibles caminos distintos.
 - Son necesarios cinco casos de prueba para lograr un 100% de cobertura de camino.
 - Sólo dos son necesarios para un 100% de cobertura de sentencia (C_0), tres son necesarios para un 100% de cobertura decisión (C_1).

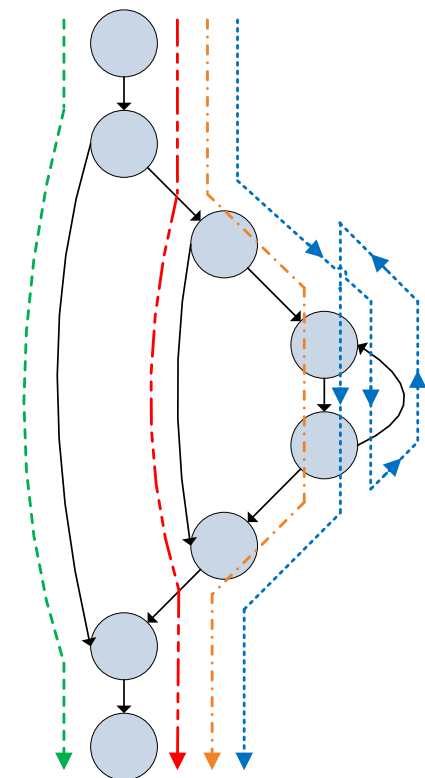


IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de camino (path coverage) – Ejemplo (2)

- El diagrama de flujo de control de la imagen, representa el segmento de programa a ser evaluado. Contiene dos sentencias **if** y un bucle en el Interior de la segunda sentencia **if**.
- Tres caminos diferentes conducen a través del diagrama de este segmento de programa logran una cobertura de decisión completa.
- Si el bucle se ejecuta dos veces son posibles cuatro caminos diferentes.
- Cada incremento en el contador del bucle añade un nuevo caso de prueba.



IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de camino (path coverage) – Conclusiones generales

- El 100% de cobertura de camino sólo se puede lograr en programas muy simples.
 - Un solo bucle puede conducir a una explosión de casos de prueba dado que todo número posible de ejecuciones de un bucle constituye un nuevo caso de prueba.
 - Teóricamente es posible un número indefinido de caminos.
- La cobertura de camino es más exhaustiva que la cobertura de sentencia y de decisión.
 - Cada posible camino a través del programa es ejecutado.
- 100% de cobertura de camino incluye 100% de cobertura de decisión y de sentencia.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Pruebas de caja blanca (White box)

- Cobertura de sentencia.
- Cobertura de decisión o rama (branch).
- Cobertura de camino (path).
- **Cobertura de condición.**

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de condición – Definición

- Se tiene en cuenta la complejidad de una condición que esté constituida por múltiples condiciones atómicas.
 - Una condición atómica no puede ser dividida en sentencias condicionales más pequeñas.
- Éste método tiene por objetivo detectar defectos que resulten de la Implementación de condiciones múltiples (condiciones combinadas).
 - Las condiciones múltiples están constituidas por condiciones atómicas que se combinan con el uso de operadores lógicos como: OR, AND, XOR, etc.
 - Ejemplo ((a>2) OR (b<6))
 - Las condiciones atómicas no contienen operadores lógicos, sólo contienen operadores relacionales y el operador NOT (=, >, <, etc.).
- Hay tres tipos de cobertura de condición.
 - Cobertura de condición simple (simple condition coverage).
 - Cobertura de condición múltiple (multiple condition coverage).
 - Mínima cobertura de condición múltiple (minimum multiple condition coverage).

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de condición simple (simple condition coverage)

- Cada sub-condición atómica de una sentencia condicional combinada debe tomar alguna vez, los valores lógicos verdadero (true) y falso (false).
- Este ejemplo se utiliza para explicar la cobertura de condición utilizando una expresión con una condición múltiple.
- Con sólo dos casos de prueba se puede lograr una cobertura de condición simple.
 - Cada sub-condición ha tomado los valores verdadero (true) y falso (false).
- Sin embargo, el resultado combinado es verdadero (true) en ambos casos.
 - true OR false = true
 - false OR true = true

Considere la siguiente condición $a > 2$ OR $b < 6$ Ejemplos de casos de prueba para cobertura de condición simple		
a = 3 (true)	b = 7 (false)	a > 2 OR b < 6 (true)
a = 1 (false)	b = 5 (true)	a > 2 OR b < 6 (true)

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de condición múltiple (multiple condition coverage)

- Todas las combinaciones que puedan ser creados utilizando permutaciones de las sub condiciones atómicas deben formar parte de las pruebas.
- Este ejemplo se utiliza para explicar la cobertura de condición utilizando una expresión con una condición múltiple.
- Con cuatro casos de prueba se puede lograr una cobertura de condición múltiple.
 - Se han creado todas las combinaciones de los valores verdadero (true) y falso (false).
 - Se han logrado todos los posibles resultados de la condición múltiple.
- El número de caso de prueba se incrementa de forma exponencial:
 - n = número de condiciones atómicas.
 - 2^n = número de casos de prueba.

Considere la siguiente condición $a > 2$ OR $b < 6$ Ejemplos de casos de prueba para cobertura de condición multiple		
a = 3 (true)	b = 7 (false)	$a > 2$ OR $b < 6$ (true)
a = 3 (true)	b = 5 (true)	$a > 2$ OR $b < 6$ (true)
a = 1 (false)	b = 5 (true)	$a > 2$ OR $b < 6$ (true)
a = 1 (false)	b = 7 (false)	$a > 2$ OR $b < 6$ (false)

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Mínima cobertura de condición múltiple (minimum multiple condition coverage)

- Todas las combinaciones que puedan ser creadas utilizando los resultados lógicos de cada sub condición cambia el resultado de la condición combinada.
- Este ejemplo se utiliza para explicar la cobertura de condición utilizando una expresión con una condición múltiple.
- Los cambios de una sub condición, cambian el resultado global para tres de los cuatro casos de prueba.
 - Sólo para el caso dos (true OR true = true) el cambio en la sub condición no resultará en un cambio en la condición global. Este caso de prueba puede ser omitido.
- El número de casos de prueba se puede reducir a un valor entre $n+1$ y $2n$

Considere la siguiente condición $a > 2$ OR $b < 6$ Ejemplos de casos de prueba para cobertura de condición multiple		
a = 3 (true)	b = 7 (false)	a > 2 OR b < 6 (true)
a = 3 (true)	b = 5 (true)	a > 2 OR b < 6 (true)
a = 1 (false)	b = 5 (true)	a > 2 OR b < 6 (true)
a = 1 (false)	b = 7 (false)	a > 2 OR b < 6 (false)

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Cobertura de condición - Conclusiones Generales

- La cobertura de condición simple es un instrumento débil para probar condiciones múltiples.
- La cobertura de condición múltiple es un método mucho mejor.
- Asegura cobertura de sentencia y decisión.
 - Sin embargo, tiene como resultado un alto número de casos de prueba: 2^n
 - La ejecución de algunas combinaciones no es posible.
 - Por ejemplo $x > 5$ AND $x < 10$ ambas sub condiciones no pueden ser falsas al mismo tiempo.
- La mínima cobertura de condición múltiple es Incluso mejor, debido a:
 - Reduce el número de casos de prueba entre $n+1$ a $2n$.
 - Las coberturas de sentencia y decisión también son cubiertas.
 - Tiene en cuenta la complejidad de las sentencias de decisión.

Todas las decisiones complejas deben ser probadas, la mínima cobertura de condición múltiple es adecuada para lograr este objetivo.

IV – Técnicas de Diseño de Pruebas

04 – Técnicas de caja blanca (white box) o basadas en la estructura

Técnicas de caja blanca - Resumen

- Los métodos de caja blanca y caja negra son métodos dinámicos, el objeto de prueba es ejecutado durante las pruebas.
- El método de caja blanca (White box) comprende:
 - Cobertura de sentencia (statement coverage).
 - Cobertura de decisión (decision coverage).
 - Cobertura de camino (path coverage).
 - Cobertura de condición (condition coverage): simple, múltiple, mínimo múltiple.
- Sólo se puede probar código existente. No se pueden hallar funciones faltantes.
- Sin embargo el código muerto o superfluo puede ser detectado con las pruebas de caja blanca. Principalmente, los métodos de caja blanca son utilizados en pruebas de bajo nivel como pruebas de componente o pruebas de integración.
- Los métodos difieren en la intensidad de las pruebas (profundidad de la prueba).
 - Dependiendo del método, el número de casos de prueba es distinto.

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

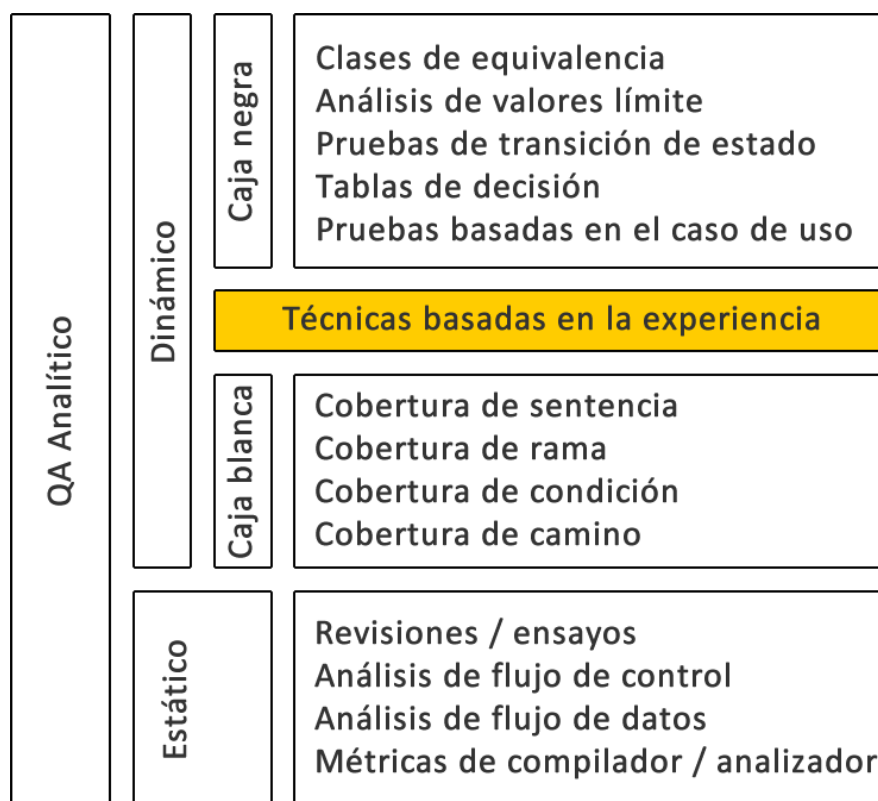
- IV/01 Pruebas en el proceso de desarrollo
- IV/02 Categorías de las técnicas de diseño de pruebas.
- IV/03 Técnicas de caja negra (black box) o basadas en la especificación.
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- **IV/05 Técnicas basadas en la experiencia.**
- IV/06 Selección de las técnicas de pruebas.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Definición

- Práctica para la creación de casos de prueba sin un claro enfoque metodológico, basada en la intuición y experiencia del probador
- Los casos de prueba se basan en la Intuición y experiencia.
 - ¿Dónde se han acumulado defectos en el pasado?
 - ¿Dónde falla el software con frecuencia?



IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Fundamentos

- Las pruebas basadas en la experiencia también se denominan pruebas intuitivas (intuitive testing) e incluye: predicción de errores (pruebas orientadas a puntos débiles) y pruebas de exploración (pruebas iterativas basadas en el conocimiento adquirido sobre el sistema).
- Principalmente se aplica con el objeto de complementar otros casos de prueba generados con mayor formalismo.
 - No cumple los criterios de un proceso de pruebas sistemático.
 - Frecuentemente produce casos de prueba adicionales que no podrían ser creados con otras prácticas, por ejemplo:
 - Problemas conocidos del pasado.
 - Conjuntos vacíos en valores de entrada.
 - Una aplicación similar ha tenido errores en estas circunstancias.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Diseño de casos de prueba

El tester debe tener experiencia y conocimientos aplicables

- **Intuición:** ¿Donde pueden existir defectos ocultos?
 - La intuición caracteriza a un buen tester.
- **Experiencia:** ¿Qué defectos se han encontrado en el pasado?
 - Conocimiento basado en la experiencia.
 - Creación de listas de errores recurrentes.
- **Conocimiento / conciencia:** ¿Cuáles son los defectos específicos esperados?
 - Detalles específicos del proyecto son incorporados.
 - ¿Cuáles son los posibles defectos ocasionados por presión de tiempo o complejidad?
 - ¿Están involucrados programadores inexpertos?

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Diseño intuitivo de casos de prueba, posibles fuentes

- Resultados de pruebas y experiencia práctica con sistemas similares.
 - Posiblemente un predecesor del software u otro sistema con funcionalidad similar.
- Experiencia de usuario.
 - Intercambio de experiencia con el sistema como usuario.
- Enfoque del despliegue.
 - ¿Qué partes del sistema serán utilizadas con mayor frecuencia?
- Problemas de desarrollo.
 - ¿Hay algún punto débil como resultado de dificultades en el proceso de desarrollo?

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Predicción de errores en la práctica

- Lista de comprobación de errores.
 - Enumerar posibles errores.
 - Factores ponderados dependientes del riesgo y probabilidad de ocurrencia.
- Diseño de caso de prueba
 - Creación de casos de prueba dirigidos a producir los errores de la lista.
 - Asignar prioridades a los casos de prueba considerando el valor de su riesgo.
- Actualizar la lista de errores durante las pruebas.
 - Procedimiento iterativo.
 - Es útil una colección estructurada de experiencia cuando se repite el procedimiento en futuros proyectos.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Pruebas exploratorias

- Es un procedimiento de diseño de casos de prueba especialmente apropiado cuando la información base se encuentra poco estructurada.
- También es útil cuando el tiempo disponible para pruebas es escaso.
- Procedimiento:
 - Revisar las partes constituyentes (Individuales / identificables) del objeto de prueba.
 - Ejecutar un número reducido de casos de prueba, exclusivamente sobre aquellas partes que deben ser probadas, aplicando predicción de errores (error guessing).
 - Analizar los resultados, desarrollar un modelo preliminar (rough model) de cómo funciona el objeto de prueba.
 - Iteración: Diseñar nuevos objetos de prueba aplicando el conocimiento adquirido recientemente.
 - Concentrarse en las áreas relevantes y explorar características adicionales del objeto de prueba.
 - Herramientas de captura pueden ser útiles para registrar las actividades de pruebas.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Pruebas exploratorias - Principios

- Seleccionar objetos pequeños y/o concentrarse en aspectos particulares del objeto de pruebas.
 - Una iteración unitaria no debería llevar más de 2 horas.
- Los resultados de una iteración constituyen la base de información para la siguiente iteración.
 - Se obtienen casos de prueba adicionales o partir de la situación particular de la prueba.
- El modelado tiene lugar durante el proceso de pruebas.
 - Se genera un modelo del objeto de prueba durante las pruebas.
 - Un objetivo de las pruebas es el refinamiento continuo del modelo.
- Preparación de pruebas adicionales
 - Con esto, el conocimiento puede ser adquirido para apoyar la elección apropiada de métodos de diseño de casos de prueba.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Diseño intuitivo de casos de prueba vs. Diseño sistemático de casos de prueba

- El diseño intuitivo de casos de prueba es un buen complemento a los enfoques sistemáticos.
 - Aún debe ser tratado como una actividad complementaria.
 - No puede dar constancia de completitud, el número de casos de prueba puede variar de forma considerable.
- Las pruebas son ejecutadas de la misma manera que las pruebas definidas de forma sistemática.
- La diferencia es la forma en la cual los casos de prueba han sido diseñados / identificados.
- A través de pruebas Intuitivas se pueden detectar defectos que no podrían ser detectados con métodos sistemáticos de prueba.

IV – Técnicas de Diseño de Pruebas

05 – Técnicas basadas en la experiencia

Técnicas basadas en la experiencia – Resumen

- Las técnicas basadas en la experiencia complementan las técnicas sistemáticas para determinar casos de prueba.
- Las técnicas basadas en la experiencia dependen en gran medida de la habilidad individual del probador.
- La predicción de errores y las pruebas exploratorias son dos de las técnicas más utilizadas de las pruebas basadas en la experiencia.

IV – Técnicas de Diseño de Pruebas

Agenda

Capítulo IV – Técnicas de Diseño de Pruebas

- IV/01 Pruebas en el proceso de desarrollo
- IV/02 Categorías de las técnicas de diseño de pruebas.
- IV/03 Técnicas de caja negra (black box) o basadas en la especificación.
- IV/04 Técnicas de caja blanca (white box) o basadas en la estructura.
- **IV/05 Técnicas basadas en la experiencia.**
- IV/06 Selección de las técnicas de pruebas.

IV – Técnicas de Diseño de Pruebas

06 – Selección de las técnicas de pruebas

Criterios para seleccionar la técnica de pruebas apropiada /1

- **Estado de la información respecto del objeto de prueba.**
 - ¿Se pueden realizar pruebas de caja blanca de alguna manera?
 - ¿Hay suficiente material de especificación para definir pruebas de caja negra, o son necesarias pruebas exploratorias para comenzar?
- **Objetivos de prueba predominantes.**
 - ¿Las pruebas funcionales han sido solicitadas de forma explícita?
 - ¿Qué pruebas no funcionales son necesarias?
 - ¿Son necesarias pruebas estructurales para lograr los objetivos del proceso de pruebas?
- **Riesgos.**
 - ¿Se espera un daño / perjuicio serio proveniente de defectos ocultos?
 - ¿Es muy alta la frecuencia de uso del objeto de prueba?
 - ¿Hay algún estándar legal o contractual, respecto de la ejecución de pruebas y cobertura de pruebas que deban ser cumplidos?

IV – Técnicas de Diseño de Pruebas

06 – Selección de las técnicas de pruebas

Criterios para seleccionar la técnica de pruebas apropiada /2

- **Precondiciones del proyecto.**

- ¿Cuánto tiempo y quien está planificando/asignando las pruebas?
- ¿Cuál es el riesgo de que el proceso de pruebas no sea finalizado según la planificación?
- ¿Qué método de desarrollo es utilizado?
- ¿Cuáles son los puntos débiles del proceso asociado al proyecto?

- **Características del objeto de prueba.**

- ¿Qué posibilidades de ser probado ofrece el objeto de prueba?
- ¿Cuál es la disponibilidad del objeto de prueba?

- **Requisitos contractuales y del cliente.**

- ¿Ha habido algún acuerdo específico entre el cliente / iniciador del proyecto respecto de los procedimientos de prueba?
- ¿Qué documentos deben ser entregados en el momento del despliegue del sistema?

IV – Técnicas de Diseño de Pruebas

06 – Selección de las técnicas de pruebas

Criterios para seleccionar la técnica de pruebas apropiada /3

- **Buenas prácticas.**
 - ¿Qué enfoques han demostrado ser apropiados para estructuras similares?
 - ¿Qué experiencias han sido adquiridas con los enfoques en el pasado?
- **Niveles de prueba.**
 - ¿En qué niveles de prueba se deben realizar pruebas?
- **Se deben aplicar criterios adicionales dependiendo de la situación específica**

IV – Técnicas de Diseño de Pruebas

06 – Selección de las técnicas de pruebas

Intereses distintos causan enfoques diferentes en el diseño de pruebas

- **Interés del jefe de proyecto:**
 - Desarrollar software de la calidad requerida.
 - Cumplir las restricciones de tiempo y presupuesto.
- **Intereses del cliente / Iniciador del proyecto:**
 - Recibir software de la más alta calidad (funcionalidad, confiabilidad, usabilidad, eficiencia, portabilidad y mantenibilidad).
 - Cumplir las restricciones de tiempo y presupuesto.
- **Intereses del jefe de pruebas:**
 - Pruebas suficientes e Intensivas. Despliegue adecuado de las técnicas necesarias desde el punto de vista del proceso de pruebas.
 - Evaluar / valorar el nivel de calidad alcanzado por el proyecto.
 - Asignar y utilizar los recursos planificados para las pruebas de forma óptima.

IV – Técnicas de Diseño de Pruebas

06 – Selección de las técnicas de pruebas

Selección de las técnicas de pruebas - Resumen

- Criterio para seleccionar el enfoque apropiado de diseño de casos de prueba:
 - Fuente de pruebas (Fuente de la información respecto de los objetos de prueba).
 - Objetivos del proceso de pruebas (Qué conclusiones se deben obtener a través de las pruebas).
 - Aspectos asociados al riesgo.
 - Estructura del proyecto / condiciones.
 - Requisitos contractuales / cliente.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- **V/01 Organización de pruebas.**
- V/02 Planificación y estimación de pruebas.
- V/03 Seguimiento y control del estado de pruebas.
- V/04 Gestión de la configuración.
- V/05 Riesgo y pruebas.
- V/06 Gestión de incidencias.

V – Gestión de Pruebas
01 – Organización de pruebas

La gestión de pruebas como parte del proceso de pruebas

- El proceso de pruebas es una actividad que cubre el proceso de desarrollo software en su totalidad.
- Las actividades propias de la gestión de pruebas son necesarias a lo largo de todo el proceso de pruebas.

Actividad

Concepción de pruebas
Planificación de pruebas
Control de pruebas
Pruebas de aceptación

Producto resultado del trabajo

Plan de pruebas maestro (estático)
Plan de pruebas (dinámico)
Informe de estado, acción de control
Liberación del producto software

V – Gestión de Pruebas
01 – Organización de pruebas

Los equipos de prueba deberían ser independientes

- **Ventajas.**

- Imparcialidad, no hay vinculación personal con el objeto de prueba.
- Se pueden cuestionar hechos respecto de la base de pruebas (test basis) y verificar las suposiciones hechas durante al diseño de pruebas.

- **Desventajas.**

- Aumenta el esfuerzo dedicado a la comunicación, presentación de conflictos del tipo "tener la última palabra".
- Desarrolladores pierden el sentido de responsabilidad

V – Gestión de Pruebas
01 – Organización de pruebas

Otras formas de conformar equipos de prueba

- Los desarrolladores prueban sus propios programas.
- Los probadores también son miembros del equipo de desarrollo.
- Los probadores también son miembros del equipo del proyecto o estructura de la organización.
- Especialistas para tareas específicas.
- Equipos de prueba externos.

V – Gestión de Pruebas

01 – Organización de pruebas

Perfiles del personal de pruebas

- El proceso de pruebas requiere personas con una amplia variedad de habilidades y cualidades.
- Se explicarán en detalle los siguientes roles asociados al proceso de pruebas:
 - Jefe de pruebas o director de pruebas (test manager).
 - Diseñador de pruebas (test designer).
 - Ingeniero de automatización de pruebas (test automation engineer).
 - Administrador de pruebas (test administrator) / Administrador del sistema de pruebas (test system administrator).
 - Probador (tester).
 - Experto técnico (technical expert).

Nota:

Se pueden especificar roles adicionales, por ejemplo administrador de base de datos, probador de carga.

V – Gestión de Pruebas

01 – Organización de pruebas

Jefe de pruebas (test leader) o director de pruebas (test manager) o coordinador de pruebas (test coordinator)

- Planifica, realiza el seguimiento y control del proyecto de pruebas.
- Competencias especiales necesarias:
 - Gestión de pruebas y calidad software.
 - Planificación y control de pruebas.
 - Experiencia como jefe de proyecto.
 - Habilidades de gestor.

V – Gestión de Pruebas
01 – Organización de pruebas

Diseñador de pruebas (test designer)

- Diseña los casos de prueba necesarios y establece el orden en el cual tendrá lugar la ejecución de los casos de prueba.
- Competencias especiales necesarias en el área de:
 - Conocimiento de desarrollo y pruebas.
 - Conocimiento de ingeniería de software.
 - Conocimiento de especificaciones técnicas.
 - Conocimiento de requisitos funcionales.

V – Gestión de Pruebas
01 – Organización de pruebas

Ingeniero de automatización de pruebas (test automation engineer)

- Evalúa las posibilidades de la automatización de pruebas y las implementa.
- Competencias especiales necesarias:
 - Experiencia como probador (tester).
 - Conocimiento técnico (know how) en el diseño y automatización de pruebas.
 - Conocimientos de programación.
 - Amplios conocimientos en el uso de las herramientas.

V – Gestión de Pruebas
01 – Organización de pruebas

Administrador del sistema de pruebas (Test system administrator)

- Prepara y opera el entorno de pruebas.
 - Es responsable de cumplir los requisitos del sistema de pruebas.
- Competencias especiales necesarias:
 - Administración de sistemas (o acceso a la administración del sistema).
 - Conocimiento de herramientas de desarrollo y pruebas.
 - Sistemas de base de datos, si aplica.
 - Redes, sí aplica.
 - Instalación y operación del sistema (por ejemplo el sistema operativo)

V – Gestión de Pruebas
01 – Organización de pruebas

Probador software (software tester)

- Ejecuta las pruebas de acuerdo con la especificación de casos de prueba.
- Competencias especiales necesarias:
 - Conocimiento básico del software.
 - Conocimiento básico de pruebas.
 - Operación y uso de herramientas de pruebas.
 - Experiencia en la ejecución de pruebas.
 - Conocimiento de los objetos de prueba.

V – Gestión de Pruebas

01 – Organización de pruebas

Experto técnico (Technical expert)

- Asiste al equipo de pruebas cuando es necesario.
- Competencias especiales necesarias:
 - Administración de bases de datos o diseño de bases de datos.
 - Experto en Interfaces de usuario.
 - Experto en redes.
- Dependiendo del tipo de problema o del entorno de pruebas puede ser necesario que expertos adicionales en pruebas formen parte del equipo de pruebas.
 - En algunas ocasiones son necesarias competencias especiales que no que no se encuentren directamente relacionadas con las pruebas, por ejemplo expertos en usabilidad, expertos en seguridad, etc.

V – Gestión de Pruebas
01 – Organización de pruebas

Competencias no técnicas (soft skills)

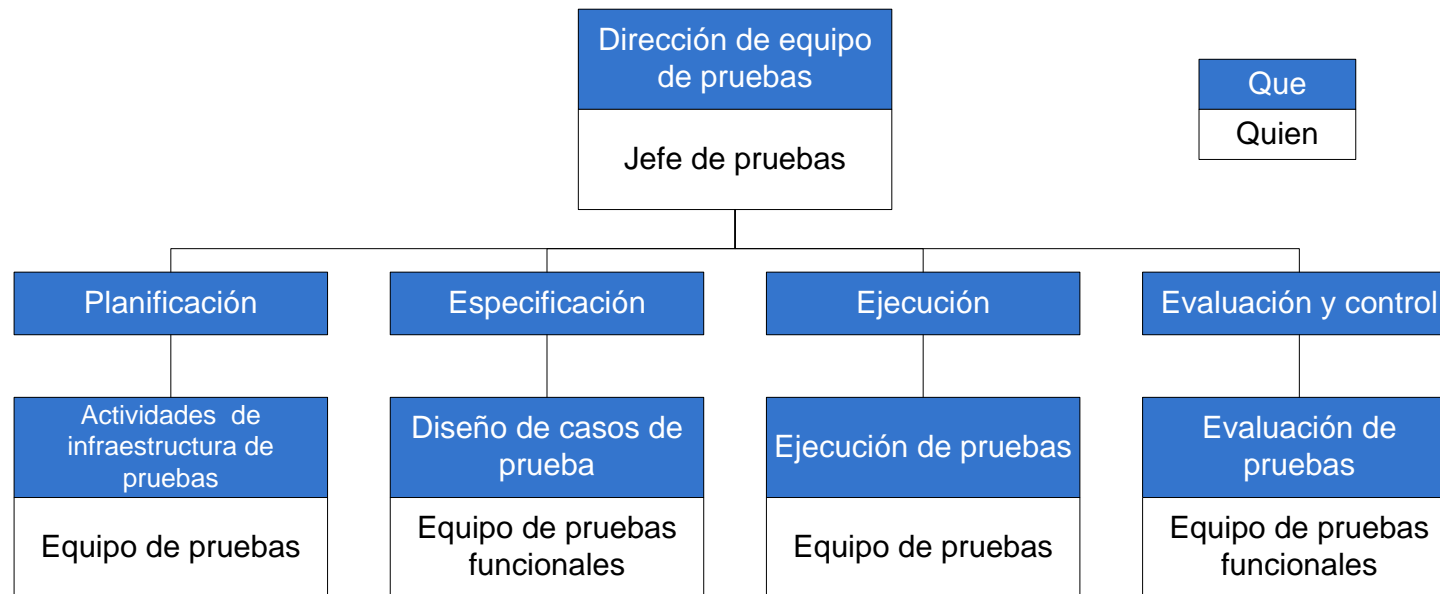
- Adicionalmente a las competencias técnicas, los miembros del equipo de pruebas requieren de las siguientes competencias y experiencias:
 - Miembros del equipo: diplomacia.
 - Disposición a preguntar sobre hechos aparentemente obvios.
 - Persistencia, fuerte personalidad.
 - Meticulosidad y creatividad.
 - Capacidad para tratar situaciones complejas.
 - Facilidad de aprendizaje

V – Gestión de Pruebas

01 – Organización de pruebas

Organización de equipos de pruebas

Utilizar una organización apropiada para cada proyecto específico.



No todos los roles los asume una persona independiente. En equipos de pruebas pequeños una persona asume múltiples roles

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Introducción

- Organización del equipo de pruebas.
- Planificación de pruebas (de acuerdo con el plan de calidad corporativo).
- Planificación de los ciclos de pruebas.
- Enfoque (Estrategia) y automatización de pruebas.
- Medición y control de pruebas.
- Introducción de un sistema de gestión de Incidencias adecuado.
- Introducción de un sistema de gestión de la configuración*.
- Informes de resultado y progreso para la dirección de la organización/compañía.

*** Esta no es una tarea particular del jefe de pruebas, la administración de la configuración es necesaria en todas las fases del desarrollo de software.**

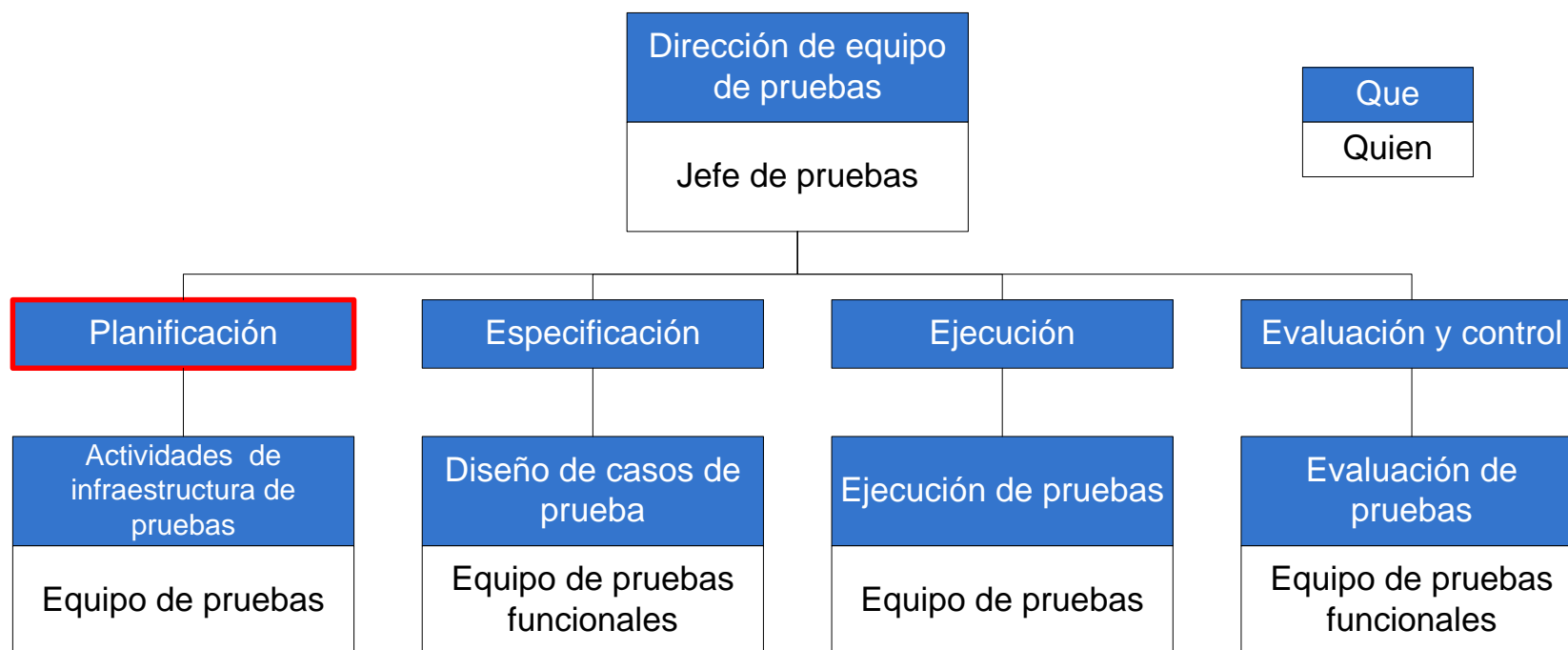
V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas - Gestión de las pruebas

- Redacción del plan de pruebas.
 - Creación de un documento que soporta métodos, recursos y plazos para las actividades del proceso de pruebas.

V – Gestión de Pruebas
01 – Organización de pruebas

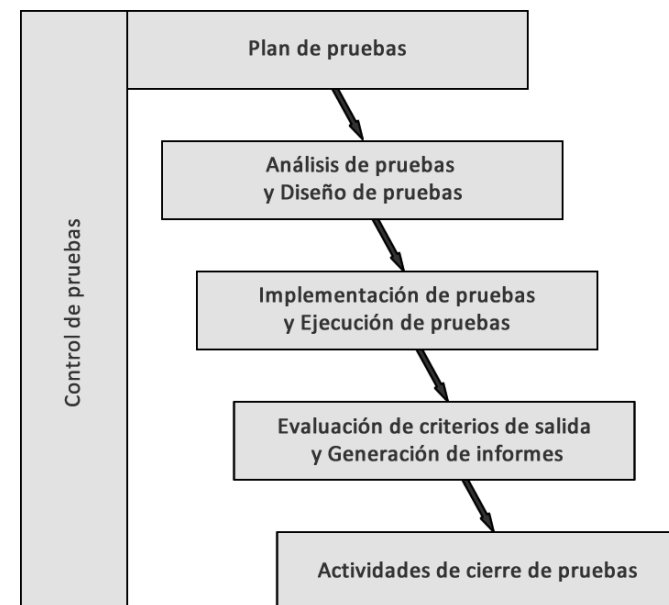
Tareas del jefe de pruebas – Planificación de pruebas, Planificación del ciclo de pruebas /1



V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Planificación de pruebas, Planificación del ciclo de pruebas /2

- **Enfoque de pruebas:** implementación de una estrategia de pruebas para un proyecto específico.
- **Ciclo de pruebas:** ciclo a través del proceso de pruebas para un objeto de prueba específico.
- **Actividades del proceso de pruebas (recordatorio):**
 - Planificación y control de pruebas.
 - Especificación de pruebas.
 - Ejecución de pruebas.
 - Evaluación y Reporte de criterios de salida
 - Cierre de pruebas



V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Planificación de pruebas, planificación del ciclo de pruebas /3

- **Planificación de pruebas.**
 - Planificación del proceso de pruebas: debe ser desarrollado en una fase temprana del proyecto. El resultado debe estar reflejado en un documento (plan de pruebas estático - static test plan).
- **Planificación del ciclo de pruebas**
 - Planificación detallada de un ciclo de pruebas: El plan de pruebas estático será detallado para describir un ciclo de prueba específico. Los detalles dependen de la situación particular del proyecto (por ejemplo progreso del desarrollo, resultados de pruebas, disponibilidad de recursos).
- **Tareas del jefe de pruebas:**
 - Iniciación, control y supervisión de pruebas y ciclos de pruebas.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Planificación de pruebas, planificación del ciclo de pruebas /4

- La planificación de las pruebas comienza al Inicio del proyecto.
- Hitos, presupuesto y prioridades de las diversas actividades del proceso de pruebas requieren ser abordados.
- Elige herramientas de pruebas y decide sobre la automatización de pruebas
 - Diferentes herramientas y grados de automatización para los diferentes niveles de pruebas.

V – Gestión de Pruebas

01 – Organización de pruebas

Tareas del jefe de pruebas – Planificación de pruebas, planificación del ciclo de pruebas /5

- Los recursos deben ser planificados.
 - Éstos son escasos y con frecuencia, deben ser asignados de forma Individual.
 - Durante los ciclos de pruebas, pueden ocurrir retrasos de tal forma que la planificación de los recursos de pruebas debe ser revisada.
- El desarrollo del proyecto debe ser tenido en cuenta.
 - A lo largo del proyecto pueden ocurrir retrasos de tal forma que puedan poner en peligro a la planificación (plazos). La planificación puede ser modificada.
 - En este caso, los casos de prueba planificados tienen que ser filtrados con el objeto de cumplir con los hitos. Ésta es, con mucha frecuencia la primera medida tomada cuando los plazos se reducen.

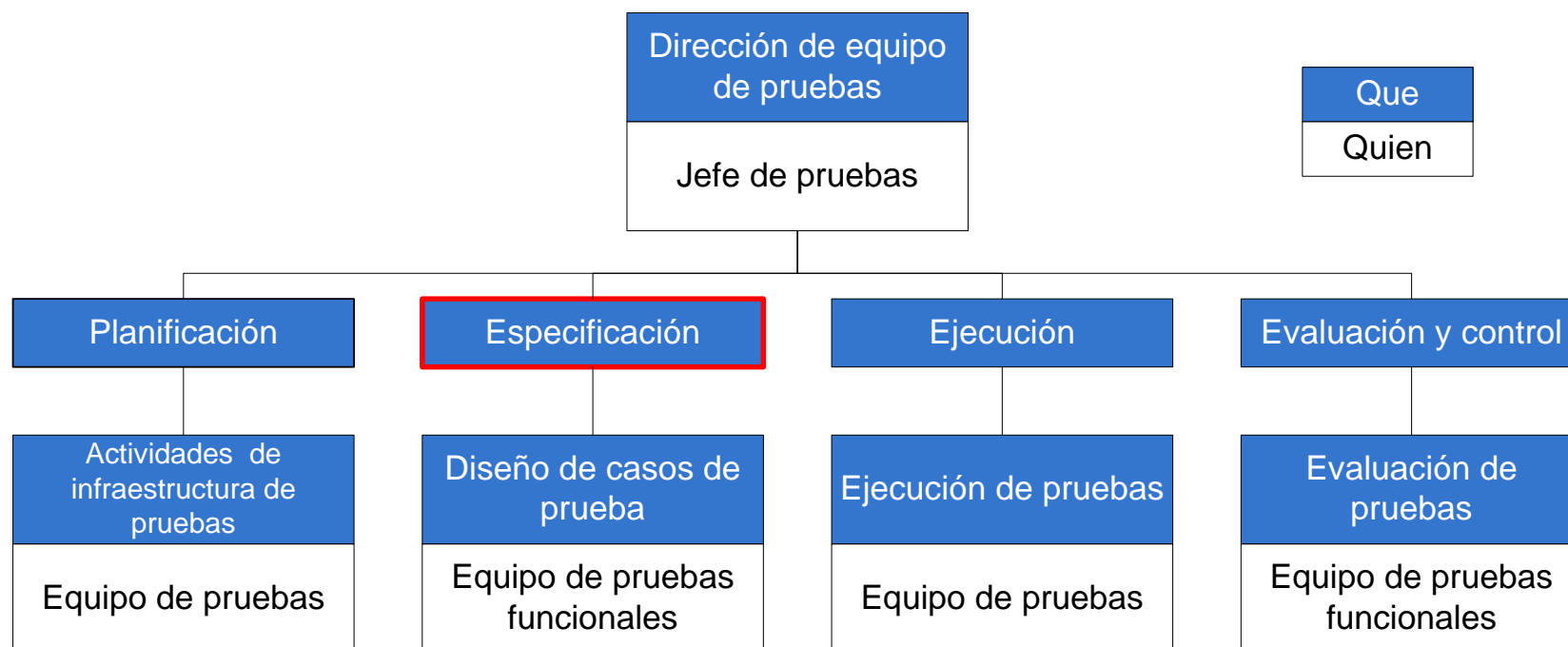
V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Planificación de pruebas, planificación del ciclo de pruebas /6

- Se debe tener en cuenta las evaluaciones de pruebas anteriores.
 - Los resultados actuales de las actividades de pruebas, pueden influir en la planificación de otras actividades. Por ejemplo, dependiendo del número de errores encontrados en un primer ciclo de pruebas, el segundo ciclo va a ser más corto o más largo.
- El control de las pruebas en ejecución se realiza usando métricas establecidos en el plan de pruebas

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Especificación de pruebas /1



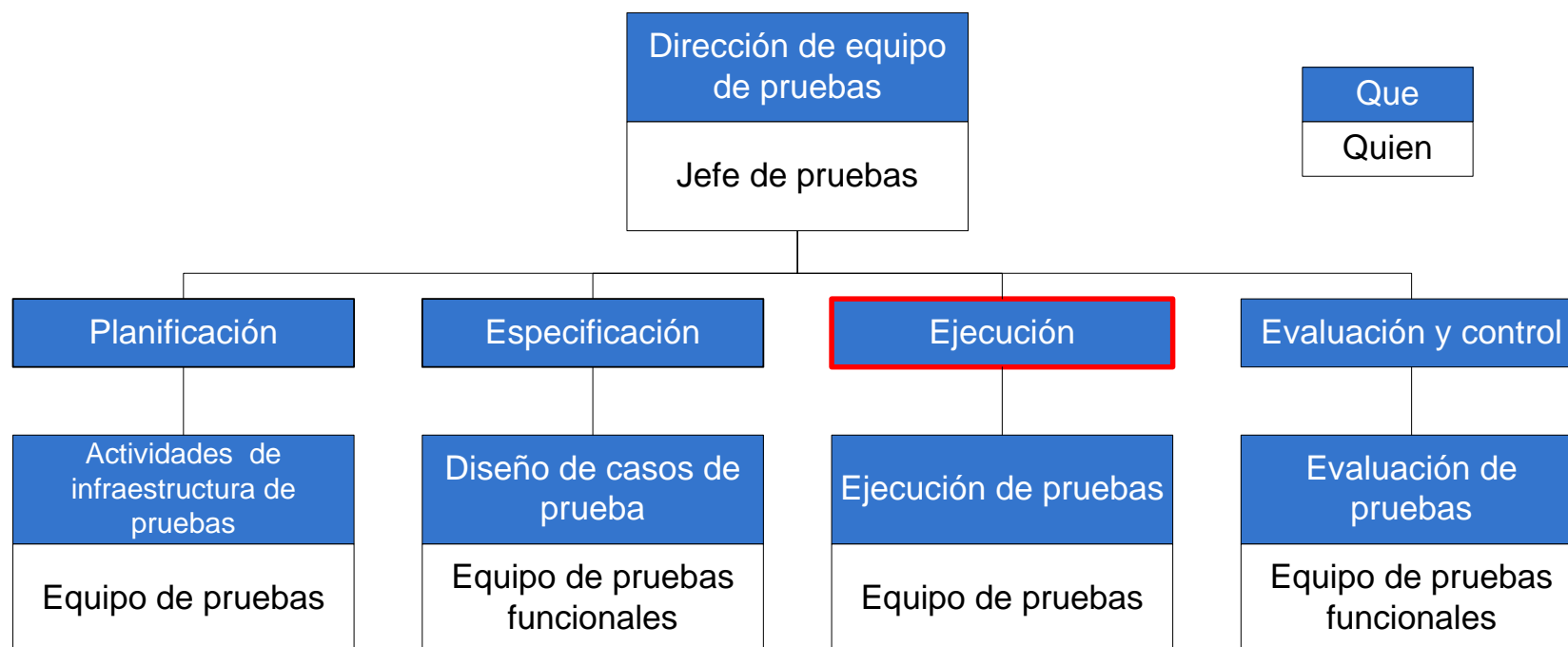
V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Especificación de pruebas /2

- El objetivo principal del proceso de pruebas es detectar la mayor cantidad de defectos relevantes con el menor esfuerzo posible.
- Todas las pruebas documentadas en el plan de pruebas son especificadas, es decir, se establece como se estructuran y como deben ser ejecutadas. Este proceso es iniciado por el jefe de pruebas.
 - Los casos de prueba están constituidos por pasos unitarios, cada paso consta de una acción y de un resultado esperado.
 - Los casos de prueba deberían ser obtenidos con la colaboración del personal que cuente con conocimiento de los requisitos funcionales del software
 - Los casos de prueba deberían ser diseñados teniendo en mente su carácter repetitivo.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Ejecución de pruebas /1



V – Gestión de Pruebas

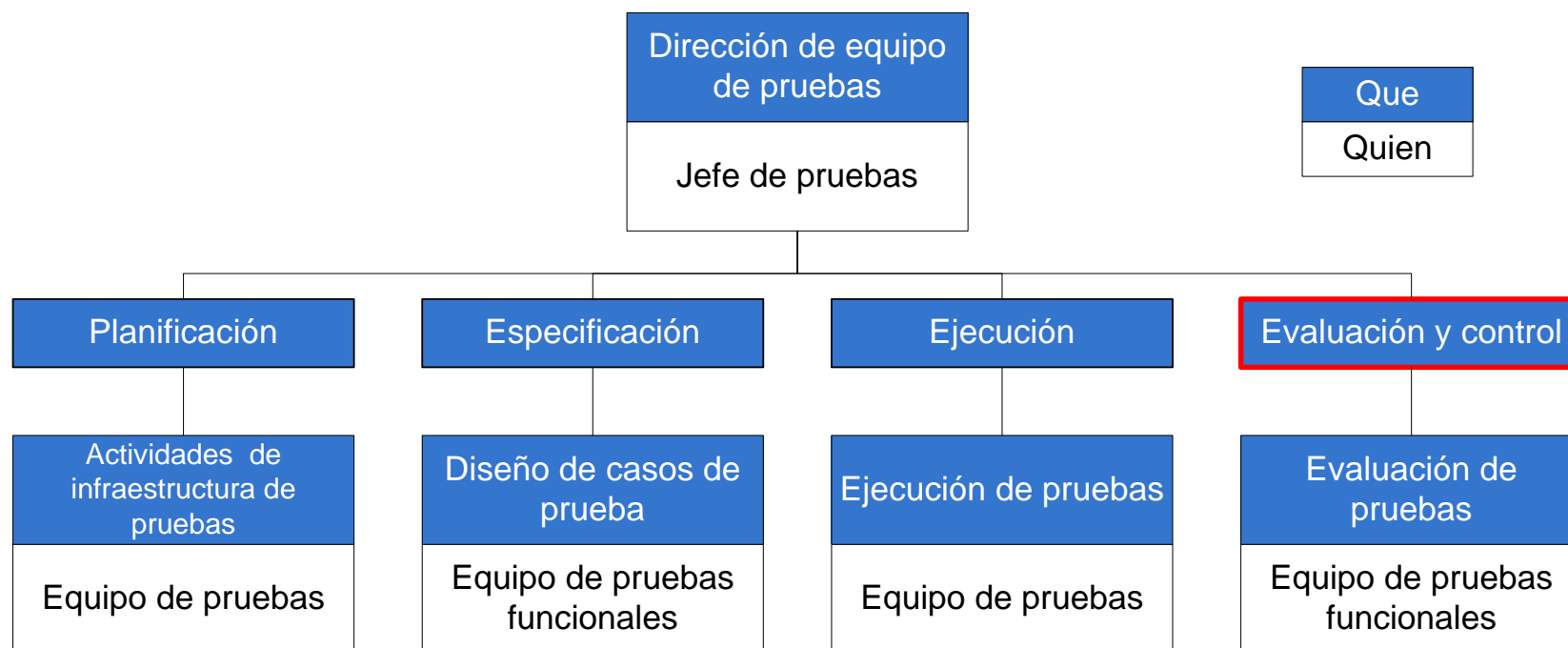
01 – Organización de pruebas

Tareas del jefe de pruebas – Ejecución de pruebas /2

- Comparación de resultados esperados y obtenidos en el proyecto.
 - Cada ciclo de pruebas requiere ser ajustado al plan de pruebas.
 - ¿Han ocurrido retrasos o cambios?
 - ¿Los resultados obtenidos se encuentran dentro del rango esperado?
 - El número de defectos detectados, tiempo utilizado en correcciones, repetición de pruebas, etc.
- Todas las desviaciones deben ser informadas y tenidas en cuenta.
 - Habitualmente las medidas correctivas deben ser tomadas de acuerdo con el plan de pruebas y las actividades de pruebas en curso, por ejemplo:
 - Ajuste de fechas para las pruebas planificadas.
 - Ajusta de recursos para la ejecución de pruebas
 - Ejecutar ciclos de pruebas adicionales, omitir ciclos de pruebas.
 - Identificar la prioridad de los ciclos de pruebas.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Evaluación y control de pruebas /1



V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Evaluación y control de pruebas /2

- La gestión de pruebas proporciona transparencia a la evolución del proceso de pruebas y aporta Indicadores a la dirección del proyecto.
- Informes generados durante la ejecución de pruebas (por ejemplo Informe de errores, Inventario organizado por tipo de defectos, estadísticas), el seguimiento de defectos e Informes al cliente son una importante fuente de información para el jefe de proyecto y la dirección de la compañía (por ejemplo como base para la planificación de recursos y plazos).
- El uso de herramientas y planillas aumentarán la calidad y pueden reducir la carga de trabajo.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del jefe de pruebas – Evaluación y control de pruebas /3

- La gestión de pruebas Incluye la aceptación de resultados del proyecto, significa: el producto debe cumplir los requisitos definidos y la especificación.
- El jefe de proyecto, de acuerdo con el Jefe de pruebas, decide respecto de la aceptación de los objetos de prueba (por ejemplo pasar a un siguiente nivel de pruebas).
- Los informes extensivos (documentación) aseguran la ejecución completa de las actividades de pruebas.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del probador *(tester) - Introducción

- Asiste la implementación del plan de pruebas.
- Lleva a cabo el diseño de casos de prueba y ejecuta las pruebas
- Revisa los casos de prueba diseñados por otros testers
- Asiste el reporte de pruebas
- Asiste la implementación de automatización de pruebas

* Probador (tester) es usado como un término genérico y puede incluir varios roles diferentes al de jefe de pruebas

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del probador (tester) /1

- **Asiste en la implementación del plan de pruebas.**
 - Especifica y comprueba planes de prueba.
 - Analiza y evalúa bases de prueba.
 - Desarrolla especificaciones de prueba
 - Formula y selecciona casos de prueba y combinaciones de datos de prueba
 - Formula resultados esperados
 - Prepara, configura y administra el entorno de pruebas (conjuntamente con los administradores de la red y de sistema)

V – Gestión de Pruebas

01 – Organización de pruebas

Tareas del probador (tester) /2

- **Ejecución de pruebas (pruebas manuales).**
 - Implementación de las pruebas a todos los niveles.
 - Ejecuta pruebas y registra resultados en un protocolo de pruebas.
 - Evalúa los resultados de las pruebas.

V – Gestión de Pruebas
01 – Organización de pruebas

Tareas del probador (tester) /3

- **Asiste en la implementación de automatización de pruebas.**
 - Crea guiones (scripts) de pruebas.
 - Crea / opera herramientas de prueba.
 - Realiza actividades de automatización de pruebas, control, captura, repetición, ejecución.
 - Analiza y evalúa resultados
- **Tratamiento después de pruebas (test post processing).**
 - Creación de protocolo de pruebas.
 - Trazar notas relativas a desviaciones.
 - Ejecutar repetición de pruebas.
 - Preparar documentación de aceptación.

V – Gestión de Pruebas

01 – Organización de pruebas

Organización de pruebas - Resumen

- La efectividad para encontrar defectos se incrementa con la independencia de pruebas del equipo de pruebas.
- El jefe de pruebas establece el equipo de pruebas en una fase temprana y:
 - Planifica y prepara todas las pruebas.
 - Establece un enfoque (estrategia) de pruebas.
 - Organiza la gestión de desviaciones y la gestión de la configuración.
 - Controla la ejecución de pruebas.
 - Evalúa los resultados de las pruebas.
- El jefe de pruebas informa a la dirección de la compañía y al jefe de proyecto.
- El probador apoya las actividades de preparación de pruebas, ejecuta pruebas, crea la documentación relativa a mensajes de desviación y resultados de pruebas. También asiste en la implementación de la automatización de pruebas.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- V/01 Organización de pruebas.
- **V/02 Planificación y estimación de pruebas.**
- V/03 Seguimiento y control del estado de pruebas.
- V/04 Gestión de la configuración.
- V/05 Riesgo y pruebas.
- V/06 Gestión de incidencias.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Actividades de planificación de pruebas.

- La planificación de pruebas es la planificación de proyectos.
 - Todas las tareas y actividades deben ser planificadas con antelación.
 - Para las distintas tareas definidas se deben asignar recursos (personal, presupuesto, herramientas, entornos de prueba, etc.).
 - Concretar las actividades de pruebas en un plan de pruebas y coordinarlas con el plan de proyecto.
 - Definir el nivel de calidad (por ejemplo profundidad de las pruebas) para los distintos niveles de pruebas.
 - La planificación de pruebas es una actividad continua, debe ser controlada de forma constante.
 - La información proveniente de las actividades de pruebas podría imponer ajustes en el plan de pruebas con el objeto de afrontar riesgos sujetos a cambios.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

La planificación de pruebas es parte de la planificación de la calidad en su conjunto.

- La planificación de pruebas es una parte importante del aseguramiento de la calidad, pero no es la única parte.
- La estructura y contenidos de un plan de calidad pueden ser encontrados en el estándar IEEE 730 (con información adicional en el estándar IEEE 983).
- Elementos de un plan de aseguramiento de la calidad de acuerdo con el estándar IEEE 730, planificación y descripción de:
 - Organización del proyecto
 - Documentos que cubren el ciclo de vida de desarrollo.
 - Estándares, métodos y convenciones. Mecanismos que aseguran que estos son cumplidos.
 - Revisiones y auditorias durante el ciclo de vida de desarrollo.
 - Proceso de pruebas.
 - Documentación de errores, soluciones.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Plan de pruebas (estático)

- Tras definir el rol del proceso de pruebas en el marco de las actividades de aseguramiento de la calidad (QA), el proceso de pruebas comienza su fase de planificación.
- El primer paso de la planificación es la creación de un plan de pruebas estático.
 - El plan de pruebas cubre todas las fases del proceso de pruebas.
 - Las reglas se fijan de acuerdo los objetivos de las pruebas, recursos, actividades de pruebas, hitos, etc.
- El plan de pruebas maestro es posteriormente ampliado con el objeto de cubrir los resultados logrados en la fase de planificación de detalle.
 - La planificación dependiente del proyecto complementa la primera versión del plan de pruebas.
 - El plan de pruebas cuenta con una extensión dinámica, que será ajustada durante el ciclo de vida del proyecto, si eso fuera necesario.
 - El estándar IEEE 829 aporta una estructura de plan de pruebas acreditada.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Plan de pruebas de acuerdo al estándar IEEE 829.

1. Introducción.
2. Supuestos.
3. Ítems de prueba.
4. Características sujetas a pruebas.
5. Características no sujetas a pruebas.
6. Enfoque.
7. Criterios de éxito / fracaso para un ítem.
8. Criterios de suspensión / reanudación.
9. Entregables de pruebas.
10. Tareas de pruebas.
11. Necesidades relativas al entorno.
12. Responsabilidades.
13. Dotación de personal y formación.
14. Calendario.
15. Riesgos y contingencias.
16. Aprobación.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Actividades de planificación de pruebas.

- La planificación de pruebas comienza al Inicio de un proyecto de desarrollo y se ajusta a lo largo del ciclo de vida del proyecto.
- La planificación de pruebas también cubre la creación y actualización del plan de pruebas. Las siguientes actividades se explican con mayor detalle:
 - Estrategia de pruebas (test strategy).
 - Planificación de recursos (resource planning).
 - Prioridad de las pruebas (priority of tests).
 - Soporte de herramientas (tool support).

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Actividades de planificación de pruebas - Estrategia de pruebas (test strategy).

- La estrategia describe los niveles de pruebas a desarrollar y la intensidad de las pruebas en esos niveles.
- La estrategia de pruebas también establece los criterios de entrada y salida para cada nivel, incluyendo las métricas para evaluar estos criterios.
- Es necesaria una estrategia de pruebas dado que probar un sistema de forma completa no es viable.
 - Probar con todas las combinaciones de datos de prueba, estados internos y restricciones temporales es prácticamente imposible.
- La valoración de riesgos ayuda a centrar la atención en aquellas áreas en que las actividades de pruebas presentan un riesgo de fallo más alto.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Actividades de planificación de pruebas - Planificación de recursos (resource planning).

- El objetivo principal de la planificación de recursos es estimar el esfuerzo de los miembros del equipo, incluyendo sus necesidades en términos de tiempo, herramientas, actividades de apoyo, etc.
 - Estas estimaciones se convierten en parte del plan de pruebas (dinámico).
- Este plan de pruebas cuenta con un calendario (time table) detallado, incluyendo hitos, asignación de personal a actividades.
 - Este plan es un Instrumento para gestionar la tarea global de la ejecución de pruebas con todas sus actividades.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Actividades de planificación de pruebas - Planificación de pruebas (test planning)

- **Gestionar tiempos:** Muchos proyectos experimentan intensos problemas de tiempo en torno a las fases finales. Esto puede conducir a decisiones sobre la reducción de actividades de pruebas u omitir pruebas de forma completa.
- **Priorizar pruebas:** Dado que la distribución de software sin haber sido probado suficientemente conlleva un alto riesgo, es necesario asignar prioridades a las actividades de pruebas. Esto debe ser realizado de tal forma que los casos de prueba más importantes sean ejecutados de forma temprana. De esta forma, partes críticas de los programas son probadas incluso en el caso en el que actividades de pruebas sean abortadas de forma prematura.
- **Selección de herramientas:** Decidir respecto de qué herramientas deben ser utilizadas para probar, si las herramientas disponibles son suficientes o si hay necesidad de herramientas adicionales.
- **Documentar:** Definir el nivel de detalle, estructura y plantillas para la documentación de pruebas.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Criterios de entrada

- Los criterios de entrada definen cuando inician las pruebas y el comienzo de un nivel de pruebas o cuando un set de pruebas está listo para ejecución.
- Criterios típicos de entrada:
 - Preparación y disponibilidad del ambiente de pruebas.
 - Preparación de herramientas de pruebas.
 - Disponibilidad de código para ser probado.
 - Disponibilidad de los datos de prueba.
 - Disponibilidad del recurso humano.
 - Probadores (testers) están listos y preparados

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Criterios de salida /1

- Los criterios de salida que indican la finalización de una fase de pruebas, deben ser establecidos para cada nivel de pruebas. Son necesarias métricas para controlar estos criterios de salida.
- **Ejemplos:**
 - Cobertura de código (code coverage).
 - Porcentaje de código (de un programa) que ha sido ejecutado.
 - Porcentaje de todas las funciones / todas las opciones de menú que han sido cubiertas.
 - Cobertura de riesgo (risk coverage).
 - Casos de prueba de una clase de riesgo predefinido (por ejemplo el nivel de riesgo más alto) han sido ejecutados con éxito en su totalidad.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Criterios de salida /2

- **Aborto de pruebas debido a razones de tiempo, costos ó calidad.**
 - Las actividades de pruebas son paralizadas / suspendidas cuando se alcanza la fecha de entrega o el presupuesto se agota. Es muy frecuente que ésta sea la realidad en proyectos, en muchas ocasiones esta circunstancia tiene un alto coste en tiempo y dinero con posterioridad.
 - Si no ha sido alcanzado un mínimo de calidad, las pruebas pueden ser suspendidas o incluso no ser iniciadas (muchos defectos críticos).

- **Tasa de detección de errores (error finding rate)**
 - El número de nuevos errores detectados cae por debajo de un valor predeterminado. Por ejemplo, Las pruebas han sido suspendidas si se han detectado menos de un error por hora.
 - Las economías del proceso de pruebas deben ser tenidas en cuenta. Más allá de una cierta tasa de detección de errores puede resultar más barato corregir errores reportados por los usuarios que buscar y eliminar errores en el proceso de pruebas.

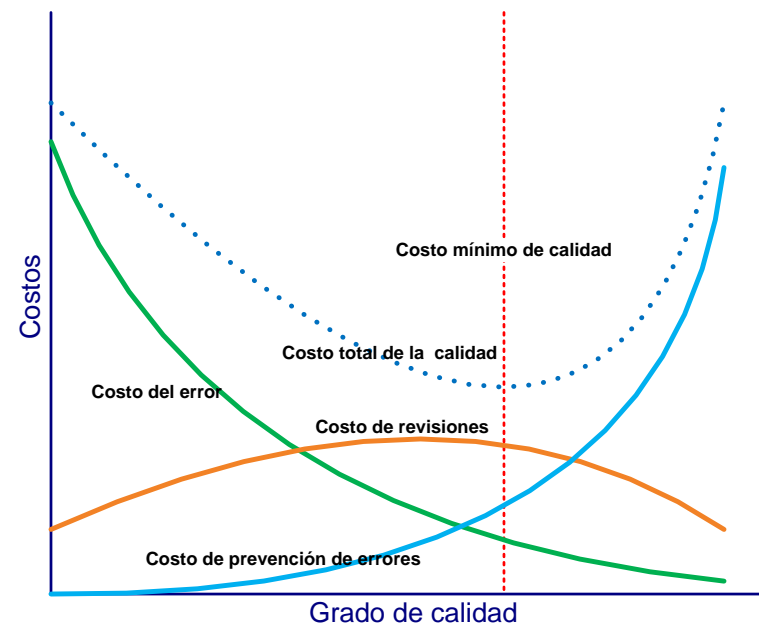
V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Criterios de salida /3

- **Economía del proceso de pruebas.**

- Un grado creciente de la calidad representa un costo más bajo de error, pero costos más altos en prevención de errores.
- Inicialmente, el costo de la revisión se incrementa, luego se reduce (las revisiones no aportan beneficios en las fases finales del proyecto).
- Inicialmente, el costo total de la calidad se reduce, luego se incrementa, los costos de la calidad son los más bajos donde la curva presenta su mínimo.
- Frecuentemente es necesario aportar un mínimo de calidad con el objeto de poder mantenerse en el negocio.
- Las pruebas deben continuar aunque los costos de prevención de errores se eleve con el fin de disminuir el costo de errores.



V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Enfoque de pruebas, estrategia de pruebas /1

El enfoque de pruebas es la implementación de la estrategia de pruebas para un proyecto específico. El enfoque de pruebas es definido y redefinido en la planificación y diseño de pruebas. Son incluidas generalmente las decisiones tomadas basadas en los objetivos del proyecto y la gestión de riesgos. Enfoques y estrategias pueden ser combinadas.

Estos son variedades de enfoques de pruebas:

- **Enfoque preventivo.**
 - Las pruebas son diseñadas tan pronto como sea posible.
- **Enfoque reactivo.**
 - Primero se diseña el sistema y el software, luego se diseñan las pruebas.
- **Enfoque analítico.**
 - Se realiza un análisis como prioridad de pruebas. Por ejemplo, pruebas basadas en riesgo.
- **Enfoque heurístico.**
 - Las pruebas son más reactivas. Por ejemplo pruebas exploratorias.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Enfoque de pruebas, estrategia de pruebas /2

Otros enfoques y estrategias de pruebas:

- **Enfoque de reutilización:** Uso de set y resultados de pruebas de proyectos anteriores con el fin de lograr avances rápidos.
- **Enfoque orientado a fallas:** Adivinación de errores.
- **Enfoque basado en listas de comprobación:** Uso de listas de comprobación de prioridades.
- **Enfoque basado en consultoría:** Expertos externos y tecnologías, guían el proceso de pruebas.
- **Enfoque de cumplimiento de procesos y estándares:** Se basa en estándares de desarrollo de software.
- **Enfoque basado en modelos:** Pruebas estocásticas basadas en información estadística, por ejemplo, información porcentual de fallas del sistema.

Más enfoques pueden ser definidos. En la práctica, los enfoques son combinados

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - factores de influencia (síntesis)

- Características del producto (por ejemplo complejidad).
- Calidad de la base de pruebas.
- Requisitos de Habilidad y seguridad del producto.
- Complejidad del proceso de desarrollo. Estabilidad de la organización, madurez del proceso utilizado.
- Personal Involucrado, restricciones temporales.
- **Métodos para estimar el esfuerzo en el proceso de pruebas.**
 - Estimación basada en expertos (estimación basada en tareas).
 - Estimación basada en analogías.
 - Estimación basada en porcentajes.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en expertos (estimación basada en tareas) /1

- **Método.**

- Identificar todas las tareas a ejecutar, normalmente utilizando un enfoque descendente (top down).
- Obtener estimaciones para cada tarea por los responsables de su ejecución o por expertos.
- Sumar todos los valores de las tareas. Incluir los factores de corrección (si hay experiencias sobre la exactitud de las estimaciones realizadas).
- Incluir elementos adicionales (buffers) con el objeto de cubrir tareas omitidas o subestimadas.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en expertos (estimación basada en tareas) /2

- **Ventajas.**

- Las actividades de estimación pueden estar estrechamente vinculadas a la planificación del proyecto.
- La estimación da origen a información detallada que puede ser controlada y ajustada a lo largo del ciclo de vida del proyecto.
- Las tareas pueden ser asignadas a grupos (por ejemplo pequeño, mediano, grande) y los esfuerzos son estimados para un representante del mismo.

- **Desventajas.**

- Este método es extensivo y costoso.
- Este método requiere de una idea clara respecto de la estrategia de pruebas y actividades de pruebas en una fase temprana del proyecto.
- La experiencia demuestra que las estimaciones son en la mayoría de los casos a la baja. Esto podría deberse a la omisión o subestimación de tareas.
- Los elementos incorporados (buffers) son recortados durante la planificación del proyecto.
- Los errores relativos a la planificación de proyecto son heredados.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en analogías /1

- **Método.**

- Clasificar las tareas de pruebas requeridas.
- Buscar un proyecto que se haya desarrollado en el pasado que contenga una tarea similar a una específica.
- Utilizar el esfuerzo real de esta tarea como base de la estimación.
- A través del uso de métricas (líneas de código, número de módulos, número de casos de prueba, etc.) como base, calcular el valor de la estimación total.
- Considerar factores de corrección.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en analogías /2

- **Ventajas.**

- El método es simple y efectivo.
- Se pueden lograr valores muy precisos para la estimación si se cuenta con suficiente experiencia.

- **Desventajas.**

- Se requiere personal con experiencia y/o información detallada respecto del proyecto actual y las tareas a estimar.
- Los criterios para la clasificación de proyectos pueden no cubrir todos los aspectos de un proyecto.
- Frecuentemente conduce a debates con la dirección respecto de la validez de la estimación.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en porcentajes /1

- **Método.**

- El esfuerzo para las actividades de pruebas se estima sobre la base de la totalidad de las actividades del proyecto.
- El valor del porcentaje (fracción) requiere ser determinado basándose en la experiencia.
- Ejemplo: Spillner, Linz habla de un porcentaje del 50% de actividades de pruebas respecto de la totalidad de las actividades del proyecto
- Este método también puede ser utilizado para otras partes del proyecto (por ejemplo estimación para los costos de gestión de proyecto, estimación del esfuerzo de pruebas para las pruebas del sistema).
- La estimación basada en porcentaje no tiene en cuenta el esfuerzo en pruebas de regresión, las cuales hacen parte importante en las pruebas de mantenimiento y las pruebas relacionadas con los cambios.

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Estimación de pruebas - Estimación basada en porcentajes /1

- **Ventajas.**

- Técnica de estimación muy simple y potente que no requiere excesiva información de entrada.

- **Desventajas.**

- No muy precisa, dado que no tiene en cuenta los hechos particulares del proyecto
- Es necesaria mucha experiencia e intuición por parte del estimador con el objeto de obtener estimaciones válidas.
- La decisión respecto del valor del porcentaje puede conducir a debates difíciles.
- Tiene en cuenta actividades que ya forman parte de las estimaciones de la planificación del proyecto, por ejemplo ¿El esfuerzo de pruebas del desarrollador forma parte de la estimación correspondiente al desarrollo o debe formar parte de las estimaciones del proceso de pruebas?

V – Gestión de Pruebas

02 – Planificación y estimación de pruebas

Resumen

- La planificación de pruebas forma parte del plan de calidad corporativo.
- El plan de pruebas es el elemento básico de toda la planificación de las actividades de pruebas. Debe ser desarrollado de forma temprana en el proyecto.
 - Plantilla de plan de pruebas: IEEE 829.
- La estimación de pruebas puede ser realizada utilizando varios métodos. Tres métodos habituales son:
 - Estimación basada en expertos (estimación basada en tareas).
 - Estimación basada en analogías.
 - Estimación basada en porcentajes.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- V/01 Organización de pruebas.
- V/02 Planificación y estimación de pruebas.
- **V/03 Seguimiento y control del estado de pruebas.**
- V/04 Gestión de la configuración.
- V/05 Riesgo y pruebas.
- V/06 Gestión de incidencias.

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Seguimiento y control del estado de pruebas

- Planificación de pruebas (test planning): Las pruebas deben ser Iniciadas.
- Seguimiento de pruebas (test monitoring): Control de las actividades de pruebas con el objeto de detectar desviaciones respecto del plan.
- Control de pruebas: Corrección del rumbo de las actividades de pruebas cuando sea necesario.
- El seguimiento debe ser realizado en base a consideraciones medibles.
 - Métricas en base a errores, por ejemplo utilizando información del sistema de gestión de incidentes.
 - Métricas en base a casos de prueba, por ejemplo utilizando Información del sistema de gestión de pruebas.
 - Métricas en base a objetos de prueba, por ejemplo utilizando Información del sistema de gestión de la configuración.
 - Métricas en base a costos, por ejemplo utilizando Información del sistema de control del proyecto.
- Los resultados obtenidos de la medición deben ser informados de forma regular.

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Informe de estado de pruebas o reporte de pruebas /1

- La Información de las actividades de pruebas se consolida en un reporte de pruebas (test reporting).
- **Ejemplo del contenido de un informe de estado de pruebas (según IEEE 829).**
 - Objetos de prueba.
 - Niveles de prueba, ciclos de prueba, período del Informe.
 - Estado de pruebas utilizando métricas, por ejemplo número de defectos documentados, número de casos de prueba ejecutados.
 - Recursos utilizados, presupuesto consumido.
 - Hitos alcanzados, por ejemplo aceptación de objetos de prueba en niveles de prueba específicos.
 - Informe de defectos (números de defectos descubiertos, número de defectos corregidos)
 - Evaluación del riesgo (nuevos riesgos, riesgos modificados por informes previos)
 - Pronóstico: Actividades planificadas para el próximo período de informe.
 - Evaluación general, estado (semáforo).

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Informe de estado de pruebas o reporte de pruebas /2

- **Frecuencia de los Informes.**

- Al inicio del proyecto, en la fase de preparación los ciclos de los informes son más largos (quincenal o mensual).
- Las fases críticas de la ejecución de pruebas requieren ciclos cortos (semanales, diarios).
- El informe de cierre de pruebas al final del proyecto.

- **Evaluación de los informes de pruebas.**

- ¿El desarrollo es apropiado?
- ¿La ejecución de las pruebas es eficaz y eficiente?
- ¿Las actividades están alineadas con los objetivos de pruebas?
- ¿Están siendo alcanzados objetivos de pruebas?
- ¿Cuál es el grado de riesgo de aquellos defectos que no han sido detectados?
- ¿Cuál es el nivel de confianza en el software basado en el estado actual de las pruebas?

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Control de pruebas

- El control de pruebas es una tarea de gestión.
 - El jefe de pruebas pertenece a las directivas del proyecto.
 - Las tareas de dirección pueden incluir otras áreas del proyecto principal, tareas ajenas al alcance de pruebas.
- Medidas correctivas como respuesta a desviaciones respecto del plan.
 - El control de pruebas incorpora todas las medidas ejecutadas durante el proceso de pruebas.
 - Ajuste de las actividades planificadas y cuando sea necesario, iniciar un nuevo ciclo de planificación en el plan de proyecto.
- Evaluación del cierre de pruebas.
 - Los criterios de salida de pruebas también son registrados con las métricas de progreso de pruebas.
 - Los criterios de salida de pruebas que hubieran sido alcanzados son documentados en el informe de pruebas para su aprobación.

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Medidas de control de pruebas

- **Provisión de mayores recursos.**
 - Más recursos humanos (sobre el tiempo, incrementar el tamaño del equipo de pruebas).
 - Más presupuesto (Incluir especialistas externos, consultores).
 - Despliegue de herramientas para la automatización de tareas.
- **Reducción del esfuerzo aplicado al trabajo.**
 - Exclusión de variaciones de casos de prueba.
 - Simplificación de objetos de prueba complejos, omisión de objetos específicos.
 - Reducción de la cantidad de datos de prueba.
 - Omisión de casos de prueba, set de prueba (juego de prueba).
- Las medidas de control de pruebas son documentadas con el objeto de informar a la dirección del proyecto o el cliente, los cambios en los riesgos para el despliegue del producto.

V – Gestión de Pruebas

03 – Seguimiento y control del estado de pruebas

Seguimiento y control del estado de pruebas - Resumen

- El seguimiento del estado de pruebas se basa en criterios medibles y aporta la Información necesaria para gestionar el proceso de pruebas.
- Las desviaciones respecto del plan requieren acciones correctivas.
- La presentación regular de informes aporta información al proyecto y a la dirección de la compañía respecto del progreso de las pruebas.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- V/01 Organización de pruebas.
- V/02 Planificación y estimación de pruebas.
- V/03 Seguimiento y control del estado de pruebas.
- **V/04 Gestión de la configuración.**
- V/05 Riesgo y pruebas.
- V/06 Gestión de incidencias.

V – Gestión de Pruebas
04 – Gestión de la configuración

Gestión de la configuración - Motivación

- Durante el desarrollo software se genera una gran cantidad de datos, información, resultados (artefactos):
 - Documentos de requisitos, especificaciones, diseño del sistema.
 - Componentes individuales, módulos integrados, sistemas completos.
- Un gran número de participantes con roles diferentes en los distintos componentes del sistema.
- La gestión de la configuración es responsable de la asignación explícita de una denominación para todos los artefactos y su administración.
 - Asignación de números de versión sucesivos.
 - Es registrado el espacio (clearance) para desarrollos posteriores.
 - Versiones antiguas son guardadas para un futuro control.
 - Es registrado el acceso a los artefactos.

V – Gestión de Pruebas
04 – Gestión de la configuración

Gestión de la configuración - Observaciones generales

- La gestión de la configuración presenta un rol de apoyo dentro de un proyecto. Todos los cambios deben ser registrados en un recurso común y comunicado haciendo uso de procesos definidos.
- Las expectativas respecto de la gestión de la configuración pueden variar de forma considerable dependiendo del tipo y alcance de proyecto, se debe desarrollar un plan de gestión de la configuración específico.
- IEEE 828 aporta un estándar para la gestión de la configuración y el plan de gestión de la configuración.
- La gestión de la configuración no es una actividad particular del proceso de pruebas, es necesaria durante todas las fases de un proyecto.
- La gestión de la configuración sin una herramienta apropiada sólo es posible en proyectos muy pequeños.

V – Gestión de Pruebas
04 – Gestión de la configuración

Gestión de la configuración - Definiciones

- Gestión de la configuración - GC (configuration management - CM) se refiere a un conjunto de medidas que complementan al desarrollo software:
 - **Gestión del cambio (change management):** sigue todas las actividades, por ejemplo cambios en el código fuente para cada solicitud de cambio.
 - **Gestión de la construcción (build management):** describe todos los pasos para crear una versión de un producto de software con el objeto de ser suministrado como un todo o subsistemas individuales.
 - **Gestión de entregas (release management):** permite la definición de versiones aisladas para cada artefacto componente de una versión completa de un producto a ser probado, entregado, etc.
 - **Gestión de versiones (versions management):** como parte de GC registra toda la información de acceso para cada artefacto: versión actual (número), último cambio, último usuario, etc.

V – Gestión de Pruebas
04 – Gestión de la configuración

Problemas tratados por la gestión de la configuración

- **¿Cuál es la versión actual?** La ambigüedad con respecto a qué versiones se corresponden puede resultar en actividades de desarrollo basadas en versiones antiguas (obsoletas) de la especificación.
- **¿Qué ha sido modificado, cuando y quien lo modificó?** Son posibles cambios concurrentes de un fichero (archivo): ¿qué cambios pueden ser sobrescritos?
- **¿Qué versión del fichero ha sido probada?** Es difícil probar y extraer una conclusión de unas pruebas cuando no se tiene conocimiento concreto de la versión de la que se trata.
- **¿Qué artefactos se corresponden?** ¿Qué versiones han sido agrupadas para crear las distintas entregas (release)?

V – Gestión de Pruebas
04 – Gestión de la configuración

Los requisitos sobre la GC conforman el punto de vista del proceso de pruebas

- Control de versiones (versión control).
 - Clasificar, guardar y recuperar diferente versiones de un objeto (V1.0, V1.1, etc.).
- Gestión de la configuración, gestión de entregas (release management).
 - Determinar y administrar toda la información en las versiones correspondientes que conforman un subsistema.
- Protocolos, comentarios y razones para los cambios realizados.
- Mantener un registro de estado.
 - Trazar defectos y cambios, registrar Informes de problemas y aportar actividades de retroalimentación (backtracking of activities).

V – Gestión de Pruebas
04 – Gestión de la configuración

Auditoria de la configuración (configuration audit)

- Se Introduce una auditoria de la configuración con el objeto de comprobar la efectividad de las actividades de la gestión de la configuración.
- La auditoria de la configuración comprobará:
 - Si todos los componentes individuales de un sistema son considerados en la gestión de la configuración.
 - Si las configuraciones individuales pueden ser identificadas correctamente.

V – Gestión de Pruebas
04 – Gestión de la configuración

Gestión de la configuración – Resumen

- La gestión de la configuración es necesaria para administrar los cambios sobre los objetos de prueba y sus respectivas versiones.
- Información de la construcción (build) y la entrega (release) es conservada con el objeto de poder reconstruir versiones antiguas.
- La gestión de la configuración se aplica al proceso de desarrollo software completo, no solamente al proceso de pruebas.
- La gestión de la configuración es posible solo con las herramientas apropiadas.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- V/01 Organización de pruebas.
- V/02 Planificación y estimación de pruebas.
- V/03 Seguimiento y control del estado de pruebas.
- V/04 Gestión de la configuración.
- **V/05 Riesgo y pruebas.**
- V/06 Gestión de incidencias.

V – Gestión de Pruebas

05 – Riesgo y pruebas

Riesgo - Definiciones

- **Riesgo**

- Un riesgo es una predicción calculada de un posible daño, la pérdida en caso de un resultado negativo, el peligro de una posible ventaja, la ganancia en el caso de un resultado positivo (oportunidad).
- El riesgo es la probabilidad de un resultado negativo (matemático), o la probabilidad de la ocurrencia de un suceso negativo multiplicada por el monto del daño económico.

- **Riesgo ("Waltzing With bears". Tom DeMarco, Timothy Lister)**

- Un caso posible en el futuro que dará lugar a un resultado no deseado (causa), este es un resultado no deseado en sí mismo (efecto).

- El riesgo asociado al proyecto y al producto deben ser tenidos en cuenta durante la planificación y el diseño de casos de prueba, cuando se prioricen casos de prueba, cuando se seleccionen métodos y durante la ejecución de pruebas.

V – Gestión de Pruebas
05 – Riesgo y pruebas

Riesgos del proyecto /1

- **Riesgos asociados a la organización (organizational risks).**
 - Capacitación y disponibilidad del personal.
 - Problemas personales entre equipos, miembros del equipo.
 - Cooperación insuficiente entre departamentos, conflictos de intereses.
 - Estimaciones no realistas de plazos del proyecto.
- **Riesgos tecnológicos (technical risk).**
 - Requisitos defectuosos, incompletos o no realistas.
 - Tecnologías, métodos y herramientas nuevas que presentan incertidumbres para el desarrollo software.
 - Déficit de calidad en productos.
 - Disponibilidad de un entorno de prueba complejo.
- **Riesgos ambientales (environmental risks).**
 - Deficiencias por factores externos en la provisión de componentes (plazos, calidad, coste)
 - Problemas de aceptación y otros inconvenientes contractuales con proveedores.
 - Acceso concerniente a recursos externos.
 - Cambios en requisitos legales.

V – Gestión de Pruebas

05 – Riesgo y pruebas

Riesgos del proyecto /2

- Los riesgos asociados al proyecto afectan al éxito del proyecto y deben ser gestionados.
- Estimación de la probabilidad y daño potencial.
- Implementar medidas apropiadas para tratar los riesgos identificados:
 - **Mitigación del riesgo:** preparación activa de medidas para reducir la probabilidad y daño potencial.
 - **Control del riesgo:** preparar las medidas necesarias en el caso en el cual el riesgo se convierte en un problema, contar con tiempo y fondos disponibles.
 - **Ignorancia del riesgo:** esperar que el riesgo no se convierta en un problema.
 - **Transferencia del riesgo:** mover el riesgo a otra área, organización.
 - **Evitar el riesgo:** (evitar situaciones de riesgo).

Cuando se analizan, manejan y mitigan los riesgos, el jefe de pruebas está siguiendo los principios establecidos para la gestión del proyecto.

IEEE Std 829 plantea esquemas para planes prueba para el manejo de riesgos y contingencias.

V – Gestión de Pruebas

05 – Riesgo y pruebas

Riesgos del producto

- Los riesgos asociados al producto son el resultado de problemas relacionados con el producto suministrado.
 - Funcionalidad insuficiente del producto suministrado.
 - Atributos no funcionales insuficientes.
 - Pobre integración de datos y calidad.
 - El producto no es idóneo para su uso previsto, por lo tanto no puede ser puesto en operación.
 - El producto provoca daños a la propiedad.
 - El producto provoca lesión o muerte accidentales.
- Las pruebas se ejecutan para reducir o evitar los riesgos asociados al producto.
 - La probabilidad de ocurrencia de un riesgo se reduce.
 - El daño potencial por la ocurrencia de un riesgo se reduce.
 - Para un potencial alto de daño, mas pruebas intensivas son necesarias

V – Gestión de Pruebas

05 – Riesgo y pruebas

Gestión de riesgos de producto /1

- Gestión de riesgos asociados al producto utilizando pruebas basadas en el riesgo.
 - Identificar, analizar y priorizar riesgos.
 - Influencia del riesgo tenido en cuenta durante la planificación de pruebas.
 - Seleccionar los métodos de pruebas para mitigar riesgos.
 - Asignar alcance de las pruebas (profundidad) de acuerdo al nivel de riesgo.
 - Adaptar el orden de ejecución de casos de prueba. Los casos de prueba importantes en primer lugar con el objeto de detectar defectos críticos de forma temprana.
 - Actualizar la lista de evaluación de riesgos (risk assessment worksheet) de forma regular.
 - Los riesgos pueden desaparecer (el proveedor ha entregado en el plazo acordado).
 - Pueden aparecer nuevos riesgos (el cliente solicita funciones adicionales).
 - Los riesgos pueden cambiar (epidemia de gripe).

V – Gestión de Pruebas

05 – Riesgo y pruebas

Gestión de riesgos de producto /1

- **Beneficios de las pruebas basadas en el riesgo:**
 - Los métodos de pruebas son seleccionados de forma particular con el objeto de mitigar los riesgos identificados.
 - El alcance de las pruebas se ocupa de los riesgos identificados.
 - El alcance del proceso de pruebas tiene en cuenta los riesgos identificados. De esta forma, el esfuerzo en el proceso de pruebas se centra en abordar la reducción del riesgo potencial.
 - Los fallos peligrosos son detectados de forma temprana, por lo tanto se hace más económica su corrección.
 - Incluso en el caso de un aborto de pruebas, se asegura que los casos de prueba más importantes han sido ejecutados (asignación de prioridades a pruebas basada en el riesgo).

V – Gestión de Pruebas

05 – Riesgo y pruebas

Riesgo y pruebas - Resumen

- Los riesgos asociados al proyecto y al producto ponen en peligro el éxito del proyecto, los riesgos deben ser gestionados.
- Los riesgos pueden ser tecnológicos, del entorno o estar asociados a la organización.
- Número de Riesgo (valor) = Probabilidad de ocurrencia por daño potencial.

V – Gestión de Pruebas

Agenda

Capítulo V – Gestión de Pruebas

- V/01 Organización de pruebas.
- V/02 Planificación y estimación de pruebas.
- V/03 Seguimiento y control del estado de pruebas.
- V/04 Gestión de la configuración.
- V/05 Riesgo y pruebas.
- **V/06 Gestión de incidencias.**

V – Gestión de Pruebas
06 – Gestión de incidencias

Detección de errores durante las pruebas

- El probador ejecuta los casos de prueba y registra los resultados.
- Posteriormente analizan las desviaciones entre los resultados esperados y los obtenidos:
 - Se identifican los fallos (los fallos pueden ocurrir en todo lugar: en documentos, en el código, en los datos de salida de un objeto de prueba, en un texto de ayuda).
- En este punto (temporal), las tareas del probador han finalizado por el momento
 - El probador espera la versión corregida del programa para ejecutar la repetición de pruebas (retest).
- Posteriormente, el seguimiento de errores se realiza utilizando un sistema de gestión de errores (sistema de gestión de errores / defectos).

V – Gestión de Pruebas
06 – Gestión de incidencias

¿Quién hace qué? /1

- Probador (tester)
 - Ejecuta los casos de prueba con el objeto de detectar errores.
 - Registra los resultados en un protocolo de pruebas.
 - Introduce los errores en un repositorio (informe de problemas).

- Jefe de pruebas (test manager).
 - Evalúa el informe de problemas.
 - Asigna prioridades a los errores (de acuerdo con la dirección del proyecto, cliente, etc.)
 - Redacta el informe de estado en función del estado actual de las labores de corrección.

V – Gestión de Pruebas
06 – Gestión de incidencias

¿Quién hace qué? /2

- Comité de Control del Cambio (CCC) (Change Control Board - CCB).
 - Decide con respecto a los cambios de requisitos y sus prioridades.
- Desarrollador (developer).
 - Analiza los fallos, localiza la causa del error.
 - Corrige la causa de error de acuerdo con la prioridad asignada.
 - Ejecuta todos los cambios aprobados.
- Todas estas tareas son ejecutadas de forma iterativa:
 - Probador (tester).
 - Jefe de pruebas (test manager).
 - Consejo de Control de Cambio (CCC)
 - Desarrollador.

V – Gestión de Pruebas
06 – Gestión de incidencias

Estructura de un informe de incidencias (informe de errores) /1

- **El Informe de incidencias describe el defecto de un error, no su causa.**
- La estructura de un informe de incidencias puede ser encontrada en el estándar IEEE 829 (Anomaly Report).
- Elementos que puede Incluir un informe de incidencias:
 - Datos del error.
 - Número único del error (puede ser generado de forma automática).
 - Objeto de prueba (nombre, versión), pasos de prueba
 - Ambiente de pruebas.
 - Nombre del autor del informe de Incidencias.
 - Fecha de la primera ocurrencia.
 - Clasificación de errores.
 - Clase de error (error class), por ejemplo crítico, menor.
 - Estado del error (error state), por ejemplo error nuevo, repetición de prueba, etc.
 - Prioridad (priority)

V – Gestión de Pruebas
06 – Gestión de incidencias

Estructura de un informe de incidencias (informe de errores) /2

- Elementos que se puede incluir un informe de incidencias:
 - Descripción.
 - Caso de prueba (aporta todos los detalles de las precondiciones).
 - Resultado erróneo, modo de fallo (usando una descripción del resultado obtenido y el resultado esperado).
 - Descripción de la desviación para facilitar su solución.
 - Severidad del error (Impacto, afectados, implicados)
 - Referencias cruzadas con informes relacionados.
 - Comentarios.
 - Acciones correctivas tomadas.
 - Registro histórico
 - Tiempo y usuario de corrección
 - Muchos sistemas siguen automáticamente los cambios en el ciclo de vida del incidente

V – Gestión de Pruebas
06 – Gestión de incidencias

Clase de error y prioridad del error

- La severidad de un error se expresa por la asignación de una clase de error.
 - Las clases de errores a utilizar pueden ser: error crítico, error mayor, error medio, error menor. Son frecuentes tres o cuatro clases de errores.
 - El criterio para la clasificación puede ser influenciado por la usabilidad del producto.
- La prioridad tiene en cuenta el efecto del error:
 - Impacto sobre la funcionalidad del programa.
 - Impacto sobre el proyecto, sobre el cliente.
 - Posibilidad de aportar una solución (corrección) inmediata al problema o en la siguiente entrega.
- La prioridad rige la urgencia de la corrección.

V – Gestión de Pruebas
06 – Gestión de incidencias

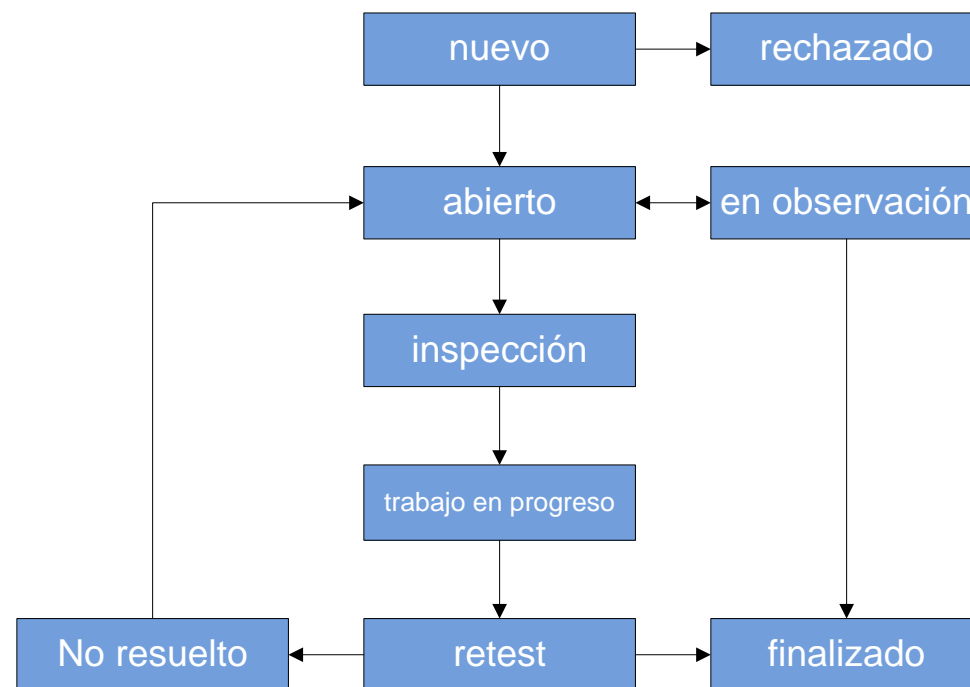
Estado de un error /1

- El estado de un error aporta información relativa al progreso o evolución del trabajo que ha sido desarrollado para este error.
- Los posibles estados de un error son, pero no están limitados a los siguientes:
 - **Nuevo (new)**: El probador ha Introducido un error en el sistema.
 - **Abierto (open)**: El Jefe de pruebas ha confirmado el informe del problema.
 - **Rechazado (rejected)**: El Jefe de pruebas ha rechazado el informe del problema.
 - **Inspección (inspection)**: El desarrollador intenta Identificar el error.
 - **En observación (surveillance)**: El error no puede ser reproducido, se encuentra bajo vigilancia.
 - **Trabajo en progreso (work in progress)**: El error es localizado y preparado para su corrección.
 - **Repetición de pruebas (retest)**: El desarrollador ha corregido la causa del error.
 - **Finalizado (finalized)**: El probador ha verificado la corrección o través de la repetición de las pruebas.
 - **No resuelto (not solved)**: El probador no ha podido verificar la corrección, el error aún está presente.

V – Gestión de Pruebas
06 – Gestión de incidencias

Estado de un error /2

Flujo de los estados típicos y transiciones de la gestión de incidentes.



El número de estados soportados por herramientas puede variar enormemente. Pueden ser 3 o pueden ser 20.

V – Gestión de Pruebas
06 – Gestión de incidencias

Estado de un error /3

- Sólo un probador puede poner un error en estado finalizado
- Normalmente el jefe de pruebas decide si un error debe ser corregido o rechazado. De forma alternativa, el comité de control del cambio puede decidir sobre la corrección de un error teniendo en cuenta el costo de la solución.
- Todos los cambios (incluidos los comentarios) deben ser registrados en el sistema de gestión de incidencias.
 - El control continuo sobre el estado de corrección de un error está asegurado.
 - Las actividades de pruebas futuras pueden ser planificadas.
 - En ocasiones, pueden ser generados casos de prueba adicionales con el objeto de localizar la causa de un fallo.

V – Gestión de Pruebas
06 – Gestión de incidencias

Análisis del informe de incidencias

- Todos los informes son analizados de manera sistemática con el fin de evaluar el estado de las actividades de corrección, el plan de conformidad del proyecto y la calidad del software.
- Típicos puntos de atención.
 - Detectar una reducción o aumento en el número de defectos nuevos durante el ciclo de vida del proyecto.
 - ¿Se detecta en un objeto de pruebas un alto número de defectos?
 - ¿Se encuentran defectos críticos aun abiertos?
 - ¿Algún defecto lleva mucho tiempo sin ser corregido?
- Los gestores de incidentes proveen una variedad de reportes de estadísticas de defectos.

V – Gestión de Pruebas
06 – Gestión de incidencias

Gestión de incidencias - Resumen

- La gestión de incidentes es la gestión de los defectos encontrados durante las pruebas.
- La gestión de incidentes es un proceso con un propio y particular flujo.
- Para el manejo de incidentes existen herramientas, que permiten cubrir las tareas para la gestión de cambios.
- La expresión gestión de desviaciones es sinónimo de gestión de incidentes.

VI – Herramientas de soporte de pruebas

Agenda

Capítulo VI – Herramientas de soporte de pruebas

- **VI/01 Tipos de herramientas de prueba.**
- VI/02 Uso efectivo de herramientas.
- VI/03 Introducción de una herramienta en una Organización.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Observaciones generales

- Las herramientas de prueba se pueden utilizar para apoyar las actividades de prueba
 - El soporte a la ejecución de la prueba se conoce como automatización de pruebas
 - Las herramientas de prueba también pueden apoyar otras actividades de prueba.
- Las herramientas de prueba se nombran según el tipo de soporte que proporcionan.
 - Las herramientas están disponibles para cada nivel del proceso de pruebas.
- En analogía con
 - CASE-Tools (Computer Aided Software Engineering), Todas las herramientas de prueba se refieren a veces como CAST-Tools (Computer Aided Software Testing)

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Entendiendo el significado y propósito de las herramientas de soporte de pruebas /1

- Herramientas que pueden utilizarse para una o más actividades de soporte de pruebas
 - Las herramientas que se utilizan directamente en las pruebas como las herramientas de ejecución pruebas, herramientas de generación de datos y herramientas de comparación de resultados.
 - Herramientas que ayudan en la gestión del proceso de pruebas, como las utilizadas para administrar las pruebas, resultados, requisitos de datos, incidencias, defectos, etc. y de información y control de ejecución de pruebas.
 - Las herramientas que se utilizan en pruebas de reconocimiento o exploración.
 - Cualquier herramienta que ayuda en la prueba (por ejemplo, una hoja de cálculo)

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Entendiendo el significado y propósito de las herramientas de soporte de pruebas /2

- Las herramientas de apoyo de pruebas pueden tener uno o más de los siguientes propósitos en función del contexto:
 - Mejorar la eficiencia de las actividades de pruebas mediante la automatización de tareas repetitivas y el apoyo a las actividades manuales como la prueba de la planificación de controles, diseño y presentación de informes.
 - Automatización de las actividades que requieren recursos importantes cuando se hace manualmente (por ejemplo, pruebas estáticas).
 - Automatización de actividades que no se puede ejecutar de forma manual (por ejemplo, pruebas de rendimiento).
 - Aumentar la fiabilidad de las pruebas (por ejemplo, mediante la automatización de las comparaciones de datos de gran tamaño o simulación del comportamiento).

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Entendiendo el significado y propósito de las herramientas de soporte de pruebas /3

- El término **MARCO DE PRUEBAS (TEST FRAMEWORK)** también se utiliza frecuentemente en la industria, por lo menos con alguno de los siguientes significados:
 - Bibliotecas reutilizables y extensibles de pruebas que se pueden utilizar para construir herramientas de prueba (también llamadas pruebas de arnés).
 - Un tipo de diseño de automatización de pruebas .
 - En general el proceso de ejecución de las pruebas.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Clasificación de herramientas de pruebas /1

- **Las herramientas para tareas especiales vs. Herramientas para suites de pruebas**
 - Herramientas individuales: Soportan una actividad de prueba particular.
 - Suite de Herramientas: Cubren varias tareas e integran varias herramientas individuales.
- **Herramientas de prueba intrusivas vs. Herramientas de prueba que no alteran el objeto de prueba**
 - **Herramientas intrusivas:** Pueden interferir en la ejecución del objeto de prueba y puede causar cambios del objeto con respecto de la ejecución del mismo en el entorno real (efecto de sonda)
 - **Depuradores** introducen puntos de ruptura y alteran el manejo de interrupciones.
 - **Drivers de prueba** proporcionan a los objetos de prueba entradas de datos simulados.
 - **La cobertura** incluye contadores que se agregan al código.
- **Esto no siempre es deseado**
 - **Durante las pruebas de rendimiento,** el objeto de prueba debe ejecutarse lo más cercano a la ejecución en el ambiente real.
 - **Durante las pruebas del sistema,** los objetos de prueba deben estar integrados en un ambiente de tiempo real.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Clasificación de herramientas de pruebas /2

- Las herramientas de prueba se pueden clasificar según varios criterios
 - Comercial, libre, de código abierto, de prueba (shareware), por la tecnología que usa.
 - Se clasifican según la actividad con la que están más estrechamente asociadas.
 - Algunas herramientas soportan una actividad, otras soportan varias actividades.
 - Paquetes de un solo proveedor que han sido diseñados para trabajar en conjunto.
- Ejemplo de herramientas que se pueden desarrollar al interior de la organización de pruebas (InHouse)
 - Hojas de cálculo en Excel.
 - SQL scripts.
 - Bases de datos para el manejo de los datos de prueba.
 - Herramientas de comparación de resultados específicos de prueba.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas /1

- **Herramientas de gestión de pruebas**

- Recolecta, clasifica y administra de casos de prueba
- Evaluación conjuntos de métricas que describen los casos de prueba
- Controla la planificación de tiempo, recursos y presupuesto.
- Crea informes de progreso, evaluación y documentación de las pruebas.
- Interconectar herramientas de ejecución de pruebas, herramientas de seguimiento de defectos y herramientas de gestión de requisitos.
- Realizar el seguimiento del objeto de prueba de acuerdo con la especificación de requisitos (consistencia).
- Gestión de versiones (gestión o administrador de la configuración).

- **Herramientas de gestión de requisitos (requerimientos).**

- Reúne los requisitos y sus atributos asociados.
- Prioriza las necesidades.
- Referencia los requisitos para los casos de prueba para comprobaciones de consistencia.
- Identifica los requisitos incompatibles y o que se están perdiendo

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas /2

- **Herramientas de manejo de Incidentes (Herramientas de seguimiento de defectos)**
 - Registra y realiza seguimiento de incidentes (defectos, fallos, anomalías, etc.).
 - Almacenar y gestionan las solicitudes de cambio.
 - Priorizan, categorizan y clasifican los defectos.
 - Evalúan y miden el grado de avance de las pruebas.
 - Muestran el flujo de trabajo para el ciclo de vida de un defecto, con el histórico de cambios de estado y responsables.

- **Herramientas de gestión de la configuración**
 - Realizan seguimiento de las versiones de los componentes:
 - Gestión de versiones de los objetos de prueba, bases de prueba, configuraciones y otras herramientas.
 - Administra las del código fuente y código del objeto.
 - Crea referencias a la gestión de pruebas, gestión de requisitos y a la gestión de cambios.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas estáticas /1

- **Herramientas de revisión.**

- Apoyo a los procesos de revisión (workflow), incluyendo listas de control, directrices, comentarios de revisión.
- Permiten documentar los resultados de la revisión.
- Permiten evaluar los resultados de la revisión.
- Proporcionan listas de control o directrices para los revisores.
- Apoyan las revisiones en línea y los equipos de pruebas ubicados en diferentes lugares.
- Proporcionar trazabilidad entre los documentos y el código fuente

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas estáticas /2

- **Herramientas de análisis estático**

- Validan el cumplimiento de estándares de codificación y codificación segura.
- Analizan de la estructura del código.
- Análisis de control de flujo, código inalcanzable (muerto), métricas para la complejidad del código (número ciclomático).
- Informa de anomalías en el flujo de datos.
- Verificación de código HTML o XML.

- **Herramientas de Modelado**

- Análisis de modelos de datos, verificación de consistencia del modelo
- Análisis de los documentos de especificaciones, diseño de modelado de objetos, diagramas de estado.
- Generación de casos de prueba basados en el modelo de software (opcional)
- **Prerequisito:** Las especificaciones se entregan en un documento con lenguaje formal.
- Por tener una relación con el proceso de desarrollo de software, es considerada comúnmente una herramienta de desarrollo.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas de especificación (funcionales) /1

- **Herramientas de diseño de prueba:** son utilizadas para generar entradas de datos de prueba, para las interfaces gráficas de usuario, modelos de diseño o en el código.
- **Herramientas para preparación de datos de prueba:** permiten manipular bases de datos, o archivos.
- Las herramientas obtienen los datos a partir de descripciones formales o definiciones de estructuras.
 - Hacen que los datos sean anónimos para garantizar la seguridad.
 - Los datos generados de forma automática a menudo tendrán que ser trabajados nuevamente de forma manual (retest).
- Las herramientas se clasifican en función de la fuente de los datos:
 - Diseño de bases de datos.
 - Código fuente.
 - Especificación de interface.
 - Especificación del objeto.
 - Robots de prueba.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas de especificación (funcionales) /2

- **Generadores de datos de prueba basados en la base de datos.**
 - Se generan los datos de las bases de datos o de archivos planos.
 - Se obtienen los datos del reconocimiento de las estructuras y los contenidos.

- **Generadores de datos de prueba basados en el código.**
 - Se generan los datos de prueba a partir del código fuente
 - No son capaces de proporcionar los valores esperados de resultado.
 - Al igual que los métodos de caja blanca, sólo pueden generar datos de prueba sobre la base del código proporcionado.
 - No pueden identificar funcionalidad faltante.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas de especificación (funcionales) /3

- **Generadores de datos de prueba basados en la Interfaz.**
 - Generación de datos de acuerdo a los parámetros de interfaz.
 - Se derivan las clases de equivalencia y valores límite directamente de los rangos definidos por los parámetros.
 - No se puede proporcionar los valores esperados de resultado, pero bien podría ser usado para pruebas de robustez.

- **Generadores de pruebas basados en las especificaciones.**
 - Generar datos de prueba directamente de los documentos de especificación.
 - La especificación de documentos debe utilizar una notación rigurosa.
 - Los documentos producidos con la ayuda de las herramientas CASE (CASE-Tools) proveen una buena base para este tipo de herramientas.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de pruebas de especificación (funcionales) /4

- **Robots de prueba**

- Pueden direccionar interfaces externas directamente del objeto de pruebas.
- Pueden aceptar o proporcionar datos, el desarrollo de la prueba se ejecuta automáticamente.
- A menudo proporcionan una función de comparación de los resultados obtenidos con los resultados esperados.
- Frecuentemente herramientas de captura y repetición son utilizadas como robots de prueba. Estas herramientas registran los pasos de ejecución de pruebas a través de la interfaz de usuario y los guarda como un archivo de comandos.
- Permiten la repetición automática de las secuencias de prueba, utilizando la secuencia de comandos grabada.
- Muy adecuado para las pruebas de regresión y las pruebas de exploración.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas para ejecución y registro de pruebas /1

- Para todos los niveles de prueba, las herramientas se pueden introducir para apoyar la ejecución de la pruebas.
- Las herramientas de ejecución de pruebas podrán abarcar las siguientes actividades:
 - Entrega de datos.
 - Recepción de datos o registro por escrito del comportamiento de datos de salida.
 - Documentación de la ejecución de pruebas.
- **Ejemplos de las herramientas de ejecución de pruebas:**
 - Herramientas de ejecución de pruebas, depurador.
 - Comparadores de prueba.
 - Pruebas de arnés, Pruebas unitarias de Framework.
 - Herramientas de medición de cobertura.
 - Herramientas de prueba de seguridad.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas para ejecución y registro de pruebas /2

- **Herramientas de ejecución de pruebas, depurador**
 - Herramienta para encontrar errores en el código del programa.
 - La secuencia de la ejecución del programa se puede interrumpir.
 - Las declaraciones individuales y las condiciones se pueden comprobar.
 - Las variables pueden ser referenciadas y definidas de forma individual.

- **Comparadores de prueba**
 - Comparan los resultados previstos y reales basadas en archivos o bases de datos de diferentes formatos.
 - Los datos a comparar son seleccionados usando funcionalidades de filtrado de datos.
 - A menudo son parte de un framework, pero también puede ser una herramienta independiente.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas para ejecución y registro de pruebas /3

- **Pruebas de arnés, Pruebas unitarias de Framework.**
 - Prueba los componentes o partes de un sistema.
 - Simulación del entorno en el que el objeto de prueba se ejecutará, a través del uso de simulación de objetos por medio de DRIVERS y/o STUBS.
 - Se trata de una réplica del entorno productivo (o una parte del mismo) y es necesario cuando restricciones de seguridad impiden el uso del entorno productivo.
 - La representación del entorno productivo debe estar lo más cerca posible.
 - Si la prueba se centra en el análisis de un componentes, puede ser también llamada prueba unitaria de framework.
 - **DRIVER**
 - Permitir el acceso al objeto de prueba cuando las interfaces no están disponibles.
 - Regula la entrada y salida de datos. Registra el progreso de la prueba.
 - Registra los resultados en tiempo real
 - Proporcionan a menudo su entorno propio sistema
 - **STUBS**
 - Simular la funcionalidad de un componente que no se encuentra disponible.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas para ejecución y registro de pruebas /4

- **Herramientas de medición de cobertura.**
 - Esta herramienta puede ser intrusiva o no intrusiva
 - Mide el porcentaje de código de una estructura que se han ejecutado para un conjunto de pruebas (set).
 - Cuenta las sentencias, las ramas, decisiones, los módulos y los llamados a funciones.

- **Herramientas de pruebas de seguridad**
 - Evalúan las características de seguridad de software
 - Evalúan la capacidad del software para proteger la confidencialidad, integridad, autenticación, autorización y disponibilidad.
 - Frecuentemente se centran en una tecnología, plataforma o propósito particular.
 - Estas herramienta son muy especializadas y son usadas por expertos.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de control y rendimiento /1

- Apoyan o automatizan las tareas de análisis de pruebas.
- **Se clasifican de acuerdo a su uso**
 - Herramientas de análisis dinámico
 - Herramientas de pruebas de rendimiento, carga y estrés.
 - Herramientas de control

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de control y rendimiento /2

- **Herramientas de análisis dinámico.**

- Encuentran defectos que son evidentes sólo cuando el software se está ejecutando.
- Encuentran dependencias de defectos.
- Encuentran pérdidas de memoria.
- Encuentran defectos en la asignación de punteros o defectos aritméticos.
- Importante para los sistemas múltiples o sistemas de sistemas.
- Normalmente se utiliza en pruebas de componentes, de integración de componentes y en el control y registro del estado interno del objeto de prueba.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte de control y rendimiento /3

- **Herramientas de pruebas de rendimiento, carga y estrés.**
 - Permiten monitoreo, medición y reporte del comportamiento del sistema bajo una variedad de condiciones simuladas de uso. Por ejemplo, número de usuarios simultáneos, frecuencia y porcentaje relativo de transacciones.
 - Permiten crear usuarios virtuales para la realización de un conjunto seleccionado de transacciones, propagándolas en varias máquinas de pruebas. Comúnmente se conocen como generadores de carga.
 - Generan una carga artificial de transacciones de usuario paralelas o tráfico de red (esto no es posible con recursos humanos).
 - Permite encontrar cuellos de botella.

- **Herramientas de control**
 - Continuamente analizan, verifican e informan sobre el uso de los recursos específicos del sistema y advierten de los posibles problemas de un servicio.

VI – Herramientas de soporte de pruebas

01 – Tipos de herramientas de prueba

Herramientas de soporte para necesidades específicas de pruebas.

- **Evaluación de la Calidad de Datos**

- Los datos son el centro de algunos proyectos
- Conversión de datos, migración de proyectos
- Aplicaciones como los almacenes de datos y sus atributos.
- Se puede variar en función de su condición crítica y el volumen
- Son necesarias herramientas para evaluar la calidad de los datos que se emplearán para verificar las conversiones de datos y reglas de migración.
- Permiten asegurarse de que los datos procesados son correctos, completos y cumplen un estándar predefinido.
- Permite establecer la dimensión de la calidad de datos (accesibilidad, credibilidad, integridad, relevancia, libre de errores, facilidad de interpretación, seguridad, etc.)
- Libre de error representa que los datos son correctos. La métrica se puede definir como de unidades de datos con error dividido por el número total de unidades.

- **Otras herramientas de pruebas existen para las pruebas de usabilidad**

- Grabadores de pantalla, es una herramienta de registro de eventos web.

VI – Herramientas de soporte de pruebas

Agenda

Capítulo VI – Herramientas de soporte de pruebas

- VI/01 Tipos de herramientas de prueba.
- **VI/02 Uso efectivo de herramientas.**
- VI/03 Introducción de una herramienta en una Organización.

VI – Herramientas de soporte de pruebas

02 – Tipos Uso efectivo de herramientas

Beneficios y riesgos potenciales de las herramientas de soporte de pruebas /1

- **El uso de herramientas de prueba ocasiona esfuerzo y costos.**
 - Proporciona la herramienta adecuada
 - Es necesario el desarrollo de habilidades para el uso de la herramienta.
 - Instalación de la herramienta en el ambiente del sistema.
 - Es necesario el ajuste de la herramienta o el ajuste de parámetros.
 - Garantiza el esfuerzo de las operaciones de administración del sistema.
 - Cambios a través del tiempo hacen necesario preparar otras pruebas.
 - Tiempo y el esfuerzo durante la operación de la herramienta.
 - Un análisis de costo/beneficio del uso de una herramienta debe realizarse antes de su implantación.
 - En algunos casos, el beneficio total sólo es evidente para el uso de herramientas en más de un proyecto.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Beneficios y riesgos potenciales de las herramientas de soporte de pruebas /2

- **Beneficios potenciales del uso de herramientas.**
 - El trabajo repetitivo se reduce.
 - Iteración de actividades idénticas.
 - Refuerzo de la consistencia y repetición.
 - Generación de métricas, por ejemplo, cobertura.
 - Facilidad de acceso a la información sobre las pruebas.
 - Gestión de datos con herramientas de prueba permite una diversidad de evaluaciones.
 - Proporcionar una mejor base de información para la toma de decisiones.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Beneficios y riesgos potenciales de las herramientas de soporte de pruebas /3

- **Los riesgos de utilizar herramientas incluyen:**
 - Funcionalidad de la herramienta de prueba no cumple las expectativas.
 - La usabilidad de la herramienta de prueba no cumple las expectativas.
 - Subestimar el tiempo y el esfuerzo necesario para alcanzar beneficios significativos y continuos por el uso de la herramienta.
 - Otros requisitos de calidad no se cumplen
 - El beneficio se había sobrestimado
 - Los costos de compra, la introducción o la operación se subestimaron.
 - Subestimar el esfuerzo necesario para mantener los datos de prueba generados por la herramienta.
 - La excesiva dependencia de la herramienta (¿las pruebas manuales serían mejor?).
 - El descuido de control de versiones de los datos de prueba generados por la herramienta.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Beneficios y riesgos potenciales de las herramientas de soporte de pruebas /3

- **Implementación incorrecta de la herramienta**

- Descuidar las relaciones y los problemas de interoperabilidad entre herramientas, tales como herramientas de gestión de requisitos, herramientas de control de versiones, herramientas de gestión de incidencias y las herramientas de otros proveedores.
- Riesgo de que los proveedores de herramientas salgan del negocio, retiró de la herramienta o venta de la herramienta a un proveedor diferente.
- Mala respuesta del proveedor de soporte, actualizaciones y soluciones de defectos.
- Riesgo de suspensión de código abierto para herramientas de este tipo.
- Falsa expectativa de que una herramienta va a resolver todos los problemas de las pruebas.
- Una herramienta no puede sustituir un proceso que no existen o compensar un procedimiento descuidado.
- Introducción de una nueva herramienta en las fases críticas del proyecto.
- Imprevistos tales como la incapacidad para soportar una nueva plataforma.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Consideraciones especiales para algunos tipos de herramienta /1

- **Herramientas de ejecución de pruebas**

- Ejecutar los objetos de prueba utilizando automáticamente scripts de prueba.
- A menudo se requiere un esfuerzo significativo para lograr importantes beneficios.
- Habilidades y conocimientos de un desarrollador para programar las secuencias de comandos (scripts) son siempre necesarios cuando se despliegan robots de prueba.
- Los resultados esperados de las pruebas tienen que ser entregados para su evaluación y la comparación automatizada, de lo contrario podría ser desperdiciado el potencial de la herramienta.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Consideraciones especiales para algunos tipos de herramienta /2

- **Enfoque de pruebas basado en datos**

- Las secuencias de comandos ejecutan las funciones del programa del objeto de prueba. El script busca datos en un archivo externo. Por ejemplo, hoja de cálculo o base de datos.
- Probadores que deseen ejecutar casos de prueba nuevos o modificados, no deben escribir nuevas secuencias de comandos, sino adaptar el archivo externo.
- Cambios en los datos o en la interfaz gráfica de usuario puede alterar la reacción del objeto de prueba, pueden producirse problemas de procesamiento.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Consideraciones especiales para algunos tipos de herramienta /3

- **Otras técnicas empleadas en las técnicas basadas en datos**
 - **Codificación de datos.**
 - Los datos son generados utilizando algoritmos basados en los parámetros configurables en tiempo de ejecución y se suministran a la aplicación.
 - Una herramienta puede usar un algoritmo que genera una muestra aleatoria de un ID de usuario. La repetición está dada en un patrón que también controla la aleatoriedad.
 - **Enfoque de pruebas basadas en palabras clave**
 - Los guiones son modulares hasta llegar al detalle de las interacciones atómicas del usuario con el objeto de prueba. Secuencias de prueba extremadamente flexibles se pueden crear sin editar los scripts.
 - Los datos de prueba y las funciones invocadas se guardan externamente. Una secuencia de comandos de control las evalúa y las lleva a las funciones particulares.
 - En el comienzo, un programador es necesario para escribir la secuencia de comandos (script).
 - Probadores serán capaces de definir las pruebas sin conocer el lenguaje de programación, sólo usan las palabras clave.
 - Problema: los datos externos necesarios crecerán rápidamente en complejidad.

Para ambas técnicas, los resultados esperados para cada prueba tienen que ser almacenados para su posterior comparación.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Consideraciones especiales para algunos tipos de herramienta /4

- Las herramientas de pruebas de rendimiento se utilizan principalmente en aplicaciones que se distribuyen y que se comunican a través de redes.
- En la mayoría de los casos, el entorno de prueba no puede ser completamente aislado y está sujeto a la influencia de factores que no se conocen en detalle en el momento de la preparación y ejecución de pruebas.
- La complejidad del medio ambiente puede hacer imposible la repetición de pruebas idénticas (los resultados son difícilmente comparables).
- En muchos casos, el conocimiento experto detallado es necesario para analizar y determinar las conclusiones correctamente.

VI – Herramientas de soporte de pruebas

02 – Uso efectivo de herramientas

Consideraciones especiales para algunos tipos de herramienta /3

- **Herramientas de análisis estático**

- Examinan el código fuente para comprobar el cumplimiento de la conformidad, por ejemplo, normas de programación.
- A menudo es necesario para preparar el código para el análisis estático.
- Un problema frecuente: una cantidad relativamente grande de indicaciones, es difícil identificar su importancia, pueden ayudar el uso de filtros.
- Una buena práctica es no sólo para limpiar los defectos, sino también las advertencias.

- **Herramientas de gestión de pruebas**

- La información debe ser abiertamente accesible.
- Una hoja de cálculo es la herramienta más utilizada por el director de pruebas para las evaluaciones y los informes.
- Los informes y las evaluaciones deben adaptarse a la organización, no al contrario.

VI – Herramientas de soporte de pruebas

Agenda

Capítulo VI – Herramientas de soporte de pruebas

- VI/01 Tipos de herramientas de prueba.
- VI/02 Uso efectivo de herramientas.
- **VI/03 Introducción de una herramienta en una Organización.**

VI – Herramientas de soporte de pruebas

03 – Introducción de una herramienta en una Organización

Selección de una herramienta para una organización /1

- Es un proceso difícil que debe ser controlado y administrado
- **Pasos hacia la introducción de herramientas**
 - **Evaluación:** Identificar las fugas o debilidades donde el proceso de prueba puede ser soportado por una herramienta de pruebas (organización, debilidades, fortalezas, etc.).
 - **Definición de Requisitos:** Las demandas de la herramienta deben ser claramente definidas, ponderadas y vinculadas a criterios medibles.
 - **Evaluación:** Realizar una lista de herramientas. Realizar pruebas de conformidad con la funcionalidad requerida y evaluar criterios de calidad, por ejemplo, licencias, soporte de proveedores, etc.
 - **Pruebas de concepto:** Identificar todos los cambios necesarios para utilizar la herramienta de manera efectiva, como por ejemplo, infraestructura, procesos. Probar si la herramienta ocasionará el efecto y soporte para el proceso de pruebas.

VI – Herramientas de soporte de pruebas

03 – Introducción de una herramienta en una Organización

Selección de una herramienta para una organización /2

- **Pasos para la implantación de la herramienta:**

- **Evaluación del proveedor:** Lista de todos los posibles candidatos con sus características claves, verificar el resultado de la evaluación y tomar una decisión final.
- **El uso de la herramienta:** Identificar los requisitos internos para el entrenamiento y formación. Lo ideal es establecer un proyecto piloto para introducir la herramienta gradualmente antes de su despliegue.
- **Evaluación de entrenamiento:** Habilidades del equipo de pruebas para orientar la formación necesaria.
- **Relación Costo/Beneficio:** Un caso de negocio concreto será la base para determinar la relación Costo/Beneficio.

VI – Herramientas de soporte de pruebas

03 – Introducción de una herramienta en una Organización

Ventajas de un proyecto piloto para la introducción de herramientas

- Conocer la herramienta en detalle con sus puntos fuertes y débiles
- Interfaz con otras herramientas en uso, la adaptación de los procesos y flujos de trabajo.
- Definición de los informes de acuerdo con las normatividad de las organizaciones.
- Evaluar si la herramienta cumple con los beneficios esperados.
- Estimar si el costo de la implementación está dentro del ámbito.
- **No se recomienda realizar despliegue sin antes ejecutar unas pruebas piloto, de lo contrario puede haber problemas de aceptación.**

VI – Herramientas de soporte de pruebas

03 – Introducción de una herramienta en una Organización

Factores de éxito para la implementación de la herramienta dentro de una organización

- Introducción paso a paso, desarrollo de forma incremental y luego implementación completa en la organización, no sólo en un proyecto o departamento.
- Hacer uso de la herramienta obligatoria para los respectivos flujos de trabajo y procesos.
- Directrices de usuario son necesarias para la implementación de herramientas.
- Los usuarios deben tener acceso a una formación adecuada, un apoyo rápido debe estar disponible para el equipo de pruebas.
- Las experiencias adquiridas desde la implementación de herramientas deben estar disponibles para todos los usuarios.
- El uso real de la herramienta debe ser objeto de seguimiento, de modo que las intervenciones necesarias puedan mejorar su aceptación.
- Recopilar de las lecciones aprendidas de todos los equipos de pruebas.

VI – Herramientas de soporte de pruebas

03 – Introducción de una herramienta en una Organización

Resumen

- **Hay una amplia gama de herramientas de prueba disponibles que abarcan diversas tareas:**
 - Herramientas de gestión de pruebas.
 - Herramientas de planificación de pruebas.
 - Herramientas de especificación de pruebas.
 - Herramientas de ejecución de pruebas.
 - Herramientas para el análisis de objetos de prueba.
 - Herramientas de apoyo a pruebas no funcionales.
- La implementación de herramientas debe llevarse a cabo sobre la base de un análisis de costo/beneficio.
- La introducción de una herramienta de prueba nueva debe ser preparada cuidadosamente con el fin de tener éxito.
- Se recomienda una implementación paso a paso, comenzando con un proyecto piloto, para el despliegue de herramientas de prueba.

AGRADECEMOS SU PARTICIPACION

LES DESEAMOS UN EXAMEN EXITOSO
PARA CONVERTIRSE EN MIEMBRO DEL

ISTQB®

COMO PROBADOR CERTIFICADO EN NIVEL BÁSICO