Técnico en

DESARROLLO DE SOFTWARE

Aseguramiento de Calidad del Desarrollo de Software

Aseguramiento de Calidad del Desarrollo de Software

Unidad III

1. Pruebas de software

Los sistemas de software son una parte creciente en nuestra vida, desde aplicaciones de negocio (bancos) hasta productos de consumo (autos).

Muchas personas han tenido experiencias con software que no trabaja como es esperado. El software que no hace su trabajo correctamente puede acarrear muchos problemas, desde pérdida de dinero, tiempo o una mala reputación, y pueden incluso causar lesiones o muerte.

Papel de las pruebas en el desarrollo de software

Las pruebas rigurosas a los sistemas y documentación pueden ayudar a reducir el riesgo de ocurrencia de problemas durante la operación y contribuyen a la calidad del sistema de software, si los defectos encontrados son corregidos antes de que el sistema sea liberado para uso operacional.

Las pruebas de software pueden ser requeridas también para conocer los requisitos contractuales o trámites legales, o las especificaciones/estándares industriales.

Pruebas y Calidad

Con la ayuda de las pruebas, es posible medir la calidad del software en términos de defectos encontrados, tanto para requisitos funcionales o no funcionales y características.

Las pruebas pueden dar confianza sobre la calidad del software si encuentra pocos o ningún defecto.

Se debe aprender de las lecciones de proyecto anteriores. Entendiendo las causas raíces de los defectos encontrados en otros proyectos, los procesos pueden ser mejorados, se debe evitar que estos efectos ocurran nuevamente y como consecuencia, mejorar la calidad de futuros sistemas. Esto es un aspecto del aseguramiento de la calidad.

Las pruebas deben ser integradas como parte de las actividades del aseguramiento de la calidad

¿Cuántas pruebas son suficientes?

En base al nivel de riesgo se determina cuantas pruebas se harán, incluyendo riesgos técnicos, de negocio y elementos del proyecto como tiempo y presupuesto.

Las pruebas deben proveer suficiente información a los involucrados para la toma de decisiones sobre la liberación del software o sistema que está siendo probado, para el paso o la entrega siguiente del desarrollo a los clientes

Una percepción común de las pruebas es que únicamente consiste en la ejecución de scripts. Esto es parte de las pruebas, pero no todas las actividades de la fase de pruebas.

Las actividades relacionadas a la fase de pruebas existen antes y después de la ejecución de pruebas: planeación y control, elección de condiciones de pruebas, diseño de casos de prueba y revisión de resultados, evaluación de criterios de salida, reportes del proceso de pruebas, finalización y clausura. También incluye la revisión de documentos.

Existen diferentes objetivos para probar:

- Encontrar defectos
- Aumentar la confianza sobre el nivel de calidad
- Prevenir defectos

El diseño de pruebas de manera temprana en el ciclo de vida puede ayudar a prevenir que defectos san introducidos en el código. La revisión de documentos (por ejemplo: Requerimientos) también puede ayudar a prevenir la aparición de defectos en el código.

La depuración y las pruebas son diferentes. Las pruebas pueden mostrar fallas causadas por defectos. La depuración es la actividad de desarrollo que identifica la causa de un defecto, reparar el código y revisa que el defecto haya sido reparado correctamente. La información subsecuente del probador asegura que la corrección ha resuelto la falla. La responsabilidad para cada actividad es muy diferente.

Principios Generales de las Pruebas

Principio 1: Las pruebas muestran la presencia de defectos.

Las pruebas pueden mostrar que los defectos están presentes, pero no pueden probar que no son defectos. Las pruebas reducen la probabilidad de mantener

defectos ocultos en el software pero, si no se encuentran defectos, no es una prueba de que esté correcto.

Principio 2: Las pruebas exhaustivas son imposibles.

Probar todo (todas las combinaciones de entradas y precondiciones) no es viable excepto para casos excepcionales. En lugar de pruebas exhaustivas, el análisis de riesgo y prioridades debería ser usado para enfocar las pruebas.

Principio 3: Pruebas tempranas.

Las actividades relacionadas a las pruebas, deben iniciar tan temprano como sea posible en el ciclo de vida de desarrollo de software, y debe enfocarse en objetivos definidos.

Principio 4: Agrupación de defectos.

Un pequeño número de módulos contienen la mayor parte de los defectos encontrados durante la pre liberación de pruebas, o es responsable de la mayor parte de las fallas operacionales.

Principio 5: Paradoja del pesticida.

Si las mismas pruebas son repetidas una y otra vez, ese conjunto de casos de prueba no encontrará más defectos.

Para evitar la paradoja del pesticida, los casos de prueba necesitan ser revisados regularmente, deben escribirse nuevos casos de prueba para probar diferentes partes del software o sistema para potenciar el hallazgo de nuevos errores.

Principio 6: Las pruebas son dependientes del contexto.

Las pruebas son diferentes en diferentes conceptos. Por ejemplo, el software crítico para un banco es probado diferente a un sitio de comercio electrónico.

Principio 7: La falacia de la ausencia de errores.

Encontrar y reparar defectos no ayuda si la construcción del sistema es inoperable y no satisface las necesidades y expectativas del cliente.

2. Ciclo de vida de las pruebas

La parte más visible de las pruebas es la ejecución, pero para qué sean efectivas y eficientes, el plan de pruebas debe incluir el tiempo esperado para la planeación de pruebas, diseño de casos de prueba, preparación para la ejecución y estatus de evaluación.

El proceso básico de pruebas consiste en las siguientes actividades principales: (ver Figura 1)

- Planeación y control
- Análisis y diseño
- Implementación y ejecución
- Evaluación de criterios de salida y reportes
- Cierre de las actividades de prueba

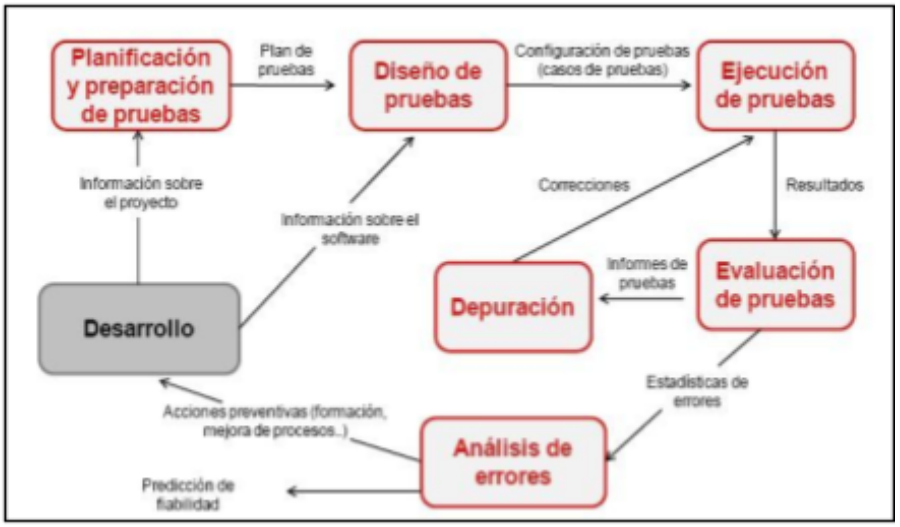

Figura 1 –Proceso Básico de Pruebas

Planeación y Control de Pruebas

La planeación de pruebas es la actividad que verifica el objetivo de las pruebas, define los objetivos y la especificación de las actividades de prueba.

El control de pruebas es la actividad que compara el progreso actual a través del plan, y reporta el estatus incluyendo las desviaciones del plan. Implica tomar acciones necesarias para conocer la misión y objetivos del proyecto.

Análisis y Diseño de las Pruebas

Es la actividad donde los objetivos generales de las pruebas son transformados en casos de prueba.

Contempla las siguientes actividades:

- Revisión de las bases de pruebas (requerimientos, arquitectura, diseño, interfaces).
- Identificar y priorizar las condiciones de pruebas basadas en el análisis de la especificación, estructura.
- Diseño y priorización de casos de prueba.

- Identificar la información necesaria para probar para dar soporte a los casos de prueba y condiciones de prueba.
- Diseño del ambiente de prueba e identificación de cualquiera herramienta o infraestructura requerida.

Implementación y Ejecución de Pruebas

Es la actividad donde los procedimientos de pruebas o scripts son especificados combinando los casos de prueba en un orden particular e incluyendo cualquier otra información necesaria para la ejecución de pruebas, el ambiente se preparara y se ejecutan las pruebas.

Se compone de las siguientes actividades:

- Desarrollo, implementación y priorización de casos de prueba. • Desarrollo y priorización de procedimientos de pruebas, creación de datos de prueba y opcionalmente codificación de scripts de prueba automatizados.
- Creación de paquetes de pruebas desde los procedimientos de prueba para la ejecución eficiente.
- Verificar que el ambiente de prueba ha sido configurado correctamente.
- Ejecución de procedimientos de pruebas ya sea manual o usando herramientas de ejecución, de acuerdo a la secuencia planeada.
- Comparación de resultados actuales contra los esperados.
- Reportar discrepancias e incidentes y analizarlos para establecer su causa (defecto en el código, en el documento, error en la ejecución).

Evaluación de Criterios de Salida y Reportes

Es la actividad donde la ejecución de pruebas se determina a través de los objetivos definidos.

Tiene las siguientes actividades:

- Revisión de bitácora de pruebas a través de los criterios de salida especificados en el plan de pruebas.
- Determinación acerca de si es necesario ejecutar más pruebas o si los criterios de salida deben ser cambiados.
- Preparar un reporte condensado de las pruebas a todos los involucrados.

Cierre de Pruebas

Recopilan información de las pruebas completadas para consolidar estadísticas, experiencias. Por ejemplo, cuando un sistema de software es liberado, un proyecto de pruebas se completa, un hito es completado.

Incluye las siguientes actividades:

Revisión de que entregables planeados han sido entregados, cierre del reporte de incidentes o levantamiento de cambios, documentación y aceptación del sistema.

- Finalización y almacenamiento del ambiente de pruebas y la infraestructura para su posterior reuso.
- Análisis de lecciones aprendidas para futuras liberaciones del proyecto y la mejora de la madurez de pruebas.

3. Estrategia de pruebas

Una estrategia de prueba del software integra los métodos de diseño de caso de pruebas del software en una serie bien planeada de pasos que desembocará en la eficaz construcción del software. La estrategia proporciona un mapa que describe los pasos que se darán como parte de la prueba, indica cuándo se

planean y cuándo se dan estos pasos, además de cuánto esfuerzo, tiempo y recursos consumirán. Por tanto, en cualquier estrategia de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas y la recolección y evaluación de los datos resultantes.

Una estrategia de prueba del software debe ser lo suficientemente flexible como para promover un enfoque personalizado. Al mismo tiempo, debe ser lo suficientemente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del proyecto.

Casos de Prueba

Un Caso de Prueba es una especificación, usualmente formal, de un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de hacer una evaluación de aspectos particulares de un elemento objeto de prueba:

- Los Casos de Prueba reflejan trazabilidad con los CU (Funcionalidad), ya que estos muestran una secuencia ordenada de eventos, al describir flujos básicos, flujos alternos, precondiciones y postcondiciones.
- Las especificaciones suplementarias de requerimientos ya que existen otras características de calidad a evaluar, además de la funcionalidad, como Usabilidad, Confiabilidad, Eficiencia, Mantenibilidad y Portabilidad.
- Las especificaciones de diseño del Sistema, ya que se debe verificar que el software fue implementado según el diseño y que los elementos arquitectónicos garantizan la calidad del software.

Esto garantiza que los procedimientos de pruebas sean compatibles con las necesidades de los usuarios/clientes. En la práctica se tiende a asumir que un

Caso de Uso en sí mismo es un Caso de Prueba y que el equipo del proyecto trabaje correctamente sobre los Casos de Usos sin planificar los Casos de Prueba. Cuando se sienta a probar los Casos de Uso, intuitivamente asume datos y procedimientos de pruebas sin que estas queden documentadas. Esto es un error ya que el Caso de Prueba extiende o amplía la información contenida en un Caso de Uso; por ejemplo, en los Casos de Uso no indicamos valores ni condiciones para las pruebas.

Los Casos de Prueba son esenciales para todas las actividades de pruebas:

- Son la base para diseñar y ejecutar los procedimientos de pruebas.
- La profundidad de las pruebas es proporcional al número de casos de pruebas.
- El diseño y desarrollo, y los recursos necesarios son gobernados por los casos de pruebas requeridos.

Si los Casos de Prueba no son correctos, la Calidad del Sistema se pone en duda y las pruebas dejan de ser confiables.

La Figura 2 muestra un modelo conceptual sobre los conceptos que están asociados a los Casos de Prueba.

En ella se puede observar que los Casos de Uso son la fuente principal de los Casos de Prueba, estos se encuentran como entregables del documento Plan de Pruebas.

Los Casos de Prueba se pueden agrupar mediante Suite de Pruebas.

Los Casos de Prueba están relacionados con el nivel de la prueba y con el tipo de prueba, la cual a su vez contiene la técnica que permite ejecutar el tipo de prueba.

Los Casos de Prueba proveen las instrucciones para el procedimiento de las pruebas.

El procedimiento de la prueba se conforma de pasos, condiciones, valores y resultados esperados y obtenidos. A su vez, el procedimiento de las pruebas puede ser automatizado a través de los script de pruebas. Todos los conceptos indicados anteriormente permiten visionar el enfoque de pruebas*: ¿qué se probará, cómo, quién, cuándo, para qué?*

Una vez ejecutados todos los Casos de Prueba, estos resultados deben reflejarse de manera global. Con ello, se establece si al validar el sistema se cumplió con los criterios de aceptación definidos con el usuario.

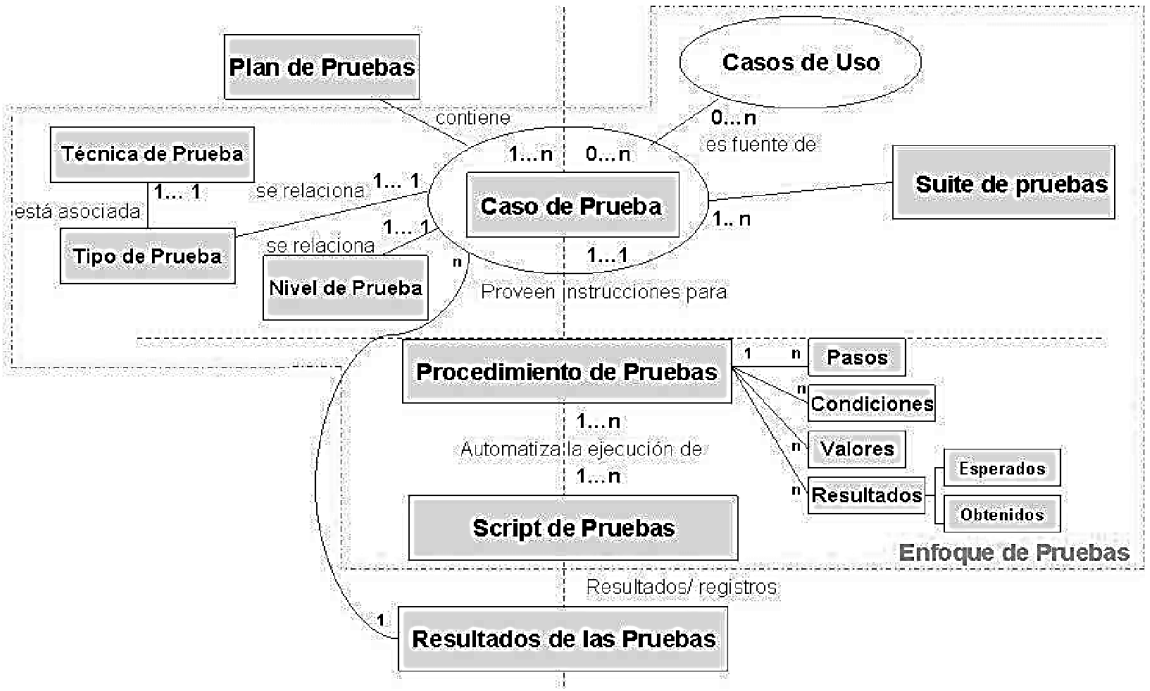

Figura 2 –Modelo conceptual asociado a Casos de Prueba

Diseño de Casos de Prueba

Un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollado para conseguir un objetivo particular o

condición de prueba como, por ejemplo, verificar el cumplimiento de un requisito específico. Para llevar a cabo un caso de prueba es necesario definir las precondiciones y postcondiciones, identificar unos valores de entrada, y conocer el comportamiento que debería tener el sistema ante dichos valores. Tras realizar ese análisis e introducir dichos datos en el sistema, se observará si su comportamiento es el previsto o no y por qué. De esta forma se determinará si el sistema ha pasado o no la prueba. De ahí su importancia durante la ejecución de pruebas.

A continuación se describirá los pasos que se realizarán en el curso para diseñar casos de pruebas.

1. Definir escenarios

Se identifican los escenarios tomando como base las narrativas de los Casos de Uso y considerando cada uno de los escenarios específicos que ocurren para cada Caso de Uso. El flujo normal, cada flujo alterno o la combinación de ellos es un escenario, que puede ser ejecutado y probado. Esto deriva que siempre el primer escenario sea el que evoca todo el flujo normal de ese Caso de Uso en particular y que la relación entre Caso de Uso y escenarios sea de uno a muchos.

Presentar gráficamente la secuencia de eventos que se plantea en cada Caso de Uso: esto permite, como lo muestra la Figura 3 abstraer los eventos que ocurren en un Caso de Uso: el flujo normal o básico y los flujos alternos, y sirve de apoyo para visualizar fácilmente las posibles combinaciones que representan un escenario ya que establece en qué punto del flujo básico ocurre y además

qué sucede después que se activa ese flujo alterno: finaliza el Caso de Uso o retorna al flujo básico.

Chequear que se hayan representado gráficamente todos los Flujos alternos con su acción de finalización o retorno.

Establecer a través de una tabla todos los escenarios asociados a un Caso de Uso, (ver Tabla 1)

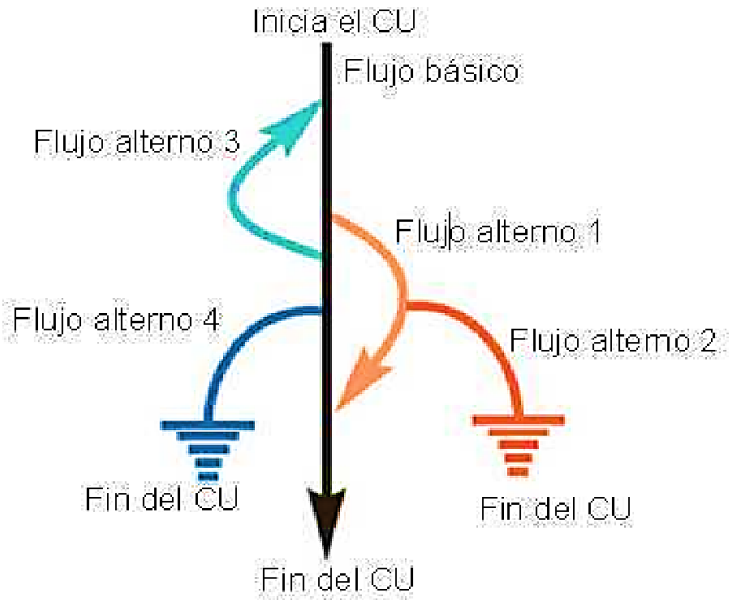

Figura 3 – Visualización de los Flujos en un CU

2. Identificar condiciones de entrada

Las condiciones de entrada son parte del dominio de valores de entrada. Se pueden identificar condiciones de entrada con estados **válidos (V)** y **no válidas (NV)**; asimismo se consideran condiciones de entrada con el estado que **no se aplica (N/A)** para un determinado escenario.

Existen los siguientes tipos de condiciones de entrada:

- Miembro de un conjunto
- Lógico
- Valor
- Rango

Nro. Escenario	Flujo Originario	Flujo alterno	Próximo alterno	Próximo alterno
Escenario 1	Flujo Básico			
Escenario 2	Flujo Básico	Flujo alterno 1		
Escenario 3	Flujo Básico	Flujo alterno 1	Flujo alterno 2	
Escenario 4	Flujo Básico	Flujo alterno 3		
Escenario 5	Flujo Básico	Flujo alterno 3	Flujo alterno 1	
Escenario 6	Flujo Básico	Flujo alterno 3	Flujo alterno 1	Flujo alterno 2
Escenario 7	Flujo Básico	Flujo alterno 4		
Escenario 8	Flujo Básico	Flujo alterno 3	Flujo alterno 4	

Tabla 1 – Escenarios por CU

Veamos un ejemplo. Considérese una aplicación bancaria, donde el usuario puede conectarse al banco por Internet y realizar una serie de operaciones bancarias. Una vez accedido al banco con las siguientes medidas de seguridad (clave de acceso y demás), la información de entrada del procedimiento que gestiona las operaciones concretas a realizar por el usuario requiere las siguientes entradas:

- *Código de banco. En blanco o número de tres dígitos. En este último caso, el primer dígito tiene que ser mayor que 1.*
- *Código de sucursal. Un número de cuatro dígitos. El primero de ellos mayor de 0.*
- *Número de cuenta. Número de cinco dígitos.*

- *Clave personal. Valor alfanumérico de cinco posiciones.*
- *Orden. Este valor se seleccionará de una lista desplegable, según la orden que se desee realizar. Puede estar en "Seleccione Orden" o una de las dos cadenas siguientes: "Talonario" o "Movimientos"*

En el caso "Talonario" el usuario recibirá un talonario de cheques, mientras que en "Movimientos" recibirá los movimientos del mes en curso. Si no se especifica este dato, el usuario recibirá los dos documentos.

A continuación se muestra una tabla con estados de las condiciones de entrada para un caso de prueba por cada escenario:

En base a la regla de generación de Casos de prueba a partir de Clases de equivalencia, se deberían tener 12 casos de prueba.

ID CP	Escenario	CONDICIONES DE ENTRADA					Resultado esperado
		Código de banco	Código de sucursal	Número de cuenta	Clave personal	Orden	
CP1	Escenario 1	V	V	V	V	V	Mensaje "Envío de talonarios"
CP2	Escenario 1	V	V	V	V	V	Mensaje "Envío de movimientos "
CP3	Escenario 1	V	V	V	V	V	lensaje "Envío de talonarios y movimientos"
CP4	Escenario 2	NV	V	V	V	V	Mensaje "Código de banco incorrecto"
CP5	Escenario 3	V	NV	V	V	V	Mensaje "Código de sucursal incorrecto"
CP6	Escenario 4	V	V	NV	V	V	Mensaje "Número de cuenta incorrecto"
CP7	Escenario 5	V	V	V	NV	V	Mensaje "Clave incorrecta"

Tabla 2 – Condiciones de Entrada

3. Definir clases de equivalencia

Pueden usarse varias técnicas para identificar los valores de los datos de entrada, la técnica de particiones o clases de equivalencias es una de ellas.

Las clases de equivalencia se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de clases de equivalencia:

- **Clases Válidas**, que representan entradas válidas al programa.
- **Clases no Válidas**, que representan valores de entrada erróneos.

Estas clases se pueden representar en una tabla. A continuación se muestra las clases de equivalencia para el caso de gestión bancaria anterior.

Sec.	Condición de Entrada	Tipo	Clases Válidas		Clases No Válidas	
			Entrada	Código	Entrada	Código
1	Código de banco	Lógico (puede estar o no)	En blanco	**CEV<01>**	Un valor no numérico	**CENV<01>**
		Si está, es Rango	200 <= Código de banco <= 999	**CEV<02>**	Código de banco < 200	**CENV<02>**
					Código de banco > 999	**CENV<03>**
2	Código de sucursal	Rango	1000 <= Código de sucursal <=9999	**CEV<03>**	Código de sucursal < 1000	**CENV<04>**
					Código de sucursal > 9999	**CENV<05>**
3	Número de cuenta	Valor	Cualquier número de 5 dígitos	**CEV<04>**	Número de más de cinco dígitos	**CENV<06>**
					Número de menos de cinco dígitos	**CENV<07>**
4	Clave personal	Valor	Cualquier cadena de caracteres alfanuméricos de 5 posiciones	**CEV<05>**	Cadena de más de cinco posiciones	**CENV<08>**
					Cadena de menos de cinco posiciones	**CENV<09>**

5	Orden	Miembro de un conjunto, con comportamiento distinto	Orden = "Seleccione Orden"	**CEV<06>**		
			Orden = "Talonario"	**CEV<07>**		
			Orden = “Movimientos”	**CEV<08>**		

Tabla 3 – Clases de Equivalencia

Realizar Casos de Prueba

En esta última etapa, se generan los casos de pruebas. Para ello, se considera como referencia la tabla de condiciones de entrada, indicando en cada caso de prueba las clases de equivalencia creadas. Por ejemplo, para el caso bancario se tendría lo siguiente:

ID CP	Clases de equivalencia	CONDICIONES DE ENTRADA					Resultado esperado
		Código de banco	Código de sucursal	Número de cuenta	Clave personal	Orden	
CP1	CEV<02>, CEV<03>, CEV<04>, CEV<05>, CEV<07>	200	1000	10000	Aaaaa	“Talonario”	Mensaje "Envío de talonarios"
CP2	CEV<01>, CEV<03>, CEV<04>, CEV<05>, CEV<08>	820	9999	99999	Zzzzz	“Movimientos ”	Mensaje "Envío de movimientos "
CP3	CEV<02>, CEV<03>, CEV<04>, CEV<05>, CEV<06>	999	1001	12345	A1b2c	“Seleccione Orden”	Mensaje "Envío de talonarios y movimientos"
CP4	CENV<01>, CEV<03>, CEV<04>, CEV<05>, CEV<07>	30A	1989	12345	1a2b3	“Seleccione Orden”	Mensaje “Código de banco incorrecto”
CP5	CENV<04>, CEV<03>, CEV<04>, CEV<05>, CEV<07>	210	999	12345	1a2b3	“Seleccione Orden”	Mensaje “Código de sucursal incorrecto”
CP6	CENV<07>, CEV<03>, CEV<04>, CEV<05>, CEV<07>	210	1989	123	1a2b3	“Seleccione Orden”	Mensaje “Número de cuenta incorrecto”
CP7	CENV<09>, CEV<03>, CEV<04>, CEV<05>, CEV<07>	210	1989	12345	“”	“Seleccione Orden”	Mensaje “Clave incorrecta”

Tabla 4 – Casos de Prueba

Informe y Seguimiento de Pruebas

De acuerdo al estándar de documentación de pruebas de software IEEE Std 829-1998, se pueden distinguir históricos, incidencias y resúmenes (ver Figura 4).

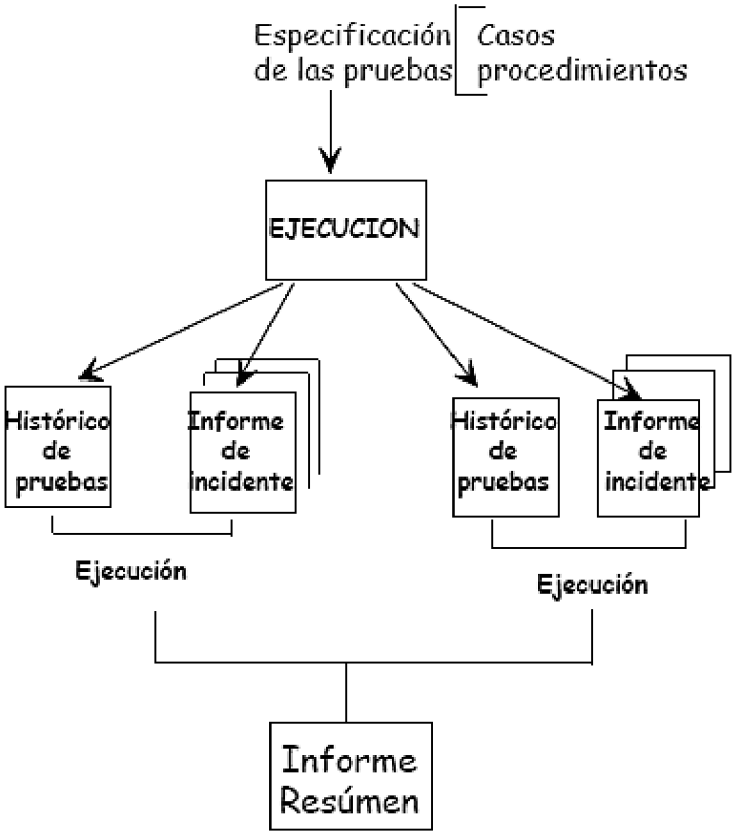

Figura 4 – Documentación Relacionada con la Ejecución de las Pruebas

El Histórico de Pruebas (Test Log) documenta todos los hechos relevantes ocurridos durante la ejecución de las pruebas. El Test Log suele tener la siguiente estructura:

- Identificador.
- Descripción de la prueba: elementos probados y entorno de la prueba.
- Anotación de datos sobre cada hecho ocurrido (incluido el comienzo y el final de la prueba)

El Informe de Incidente (Test Incident Report) documenta cada incidente (por ejemplo, una interrupción en las pruebas debido a un corte del fluido eléctrico, bloqueo del teclado) ocurrido en la prueba y que requiera de una posterior investigación. El Informe de Incidente, debe tener la siguiente estructura:

- Identificador.
- Resumen del incidente.
- Descripción de datos objetivos (fecha/hora, entradas, resultados esperados)
- Impacto que tendrá sobre las pruebas.

El Informe Resumen de Pruebas (Test Summary Report) resume los resultados de las actividades de prueba (las señaladas en el propio informe) y aporta una evaluación del software basada en dichos resultados. El Informe Resumen de Pruebas deberá tener la siguiente estructura:

- Identificador.
- Resumen de la evaluación de los elementos probados.
- Variaciones del software respecto a su especificación de diseño, así como las variaciones en las pruebas.
- Valoración de la extensión de la prueba (cobertura lógica, funcional, de requisitos).
- Resumen de los resultados obtenidos en las pruebas.
- Evaluación de cada elemento software sometido a prueba (evaluación general del software incluyendo las limitaciones del mismo).
- Firmas y aprobaciones de quienes deban supervisar el informe.

Relación entre las Pruebas y la Depuración

Depuración es el proceso de analizar y corregir los efectos que sospecha que contiene el software, ver Figura 5.

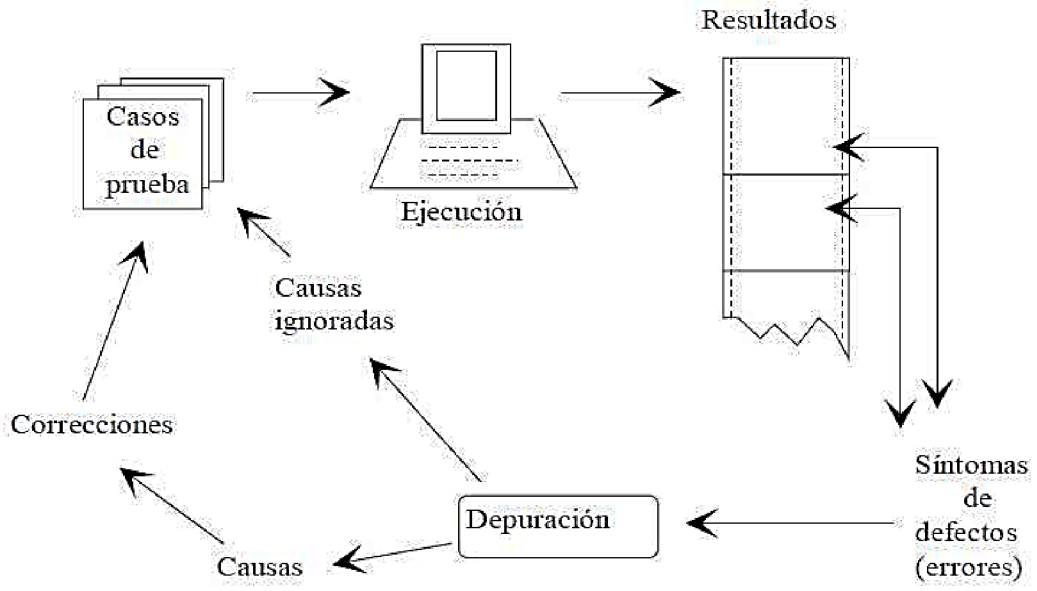

Figura 5 – Proceso de Depuración

Durante la Depuración es necesario tener en cuenta lo siguiente, para el proceso de localización del error:

- Analizar la información y pensar (analizar bien, en lugar de aplicar un enfoque aleatorio de búsqueda del efecto).
- Al llegar a un punto muerto, pasar a otra cosa (refresca la mente).
- Al llegar a un punto muerto, describir el problema a otra persona (el simple hecho de describir el problema a alguien puede ayudar).
- Usar herramientas de depuración sólo como recurso secundario (no sustituir el análisis mental).
- No experimentar cambiando el programa (no sé qué está mal, así que cambiaré esto y veré lo que sucede No).
- Se debe atacar los errores individualmente.

- Se debe fijar la atención también en los datos (no sólo en la lógica del programa).

Durante el proceso de corrección del error, es necesario considerar las siguientes recomendaciones:

- Donde hay un defecto, suele haber más (lo dice la experiencia).
- Debe fijarse el defecto, no sus síntomas (no debemos enmascarar síntomas, sino corregir el efecto).
- La probabilidad de corregir perfectamente un defecto no es del 100% (cuando se corrige, hay que probarlo).
- Cuidado con crear nuevos defectos (a corregir defectos se producen otro nuevos).
- La corrección debe situarnos temporalmente en la fase de diseño (hay que retocar desde el comienzo, no sólo el código).
- Cambiar el código fuente, no el código objeto.

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.