Ingeniería del Software I

Guía de Estudio
Gestión de Calidad

Docentes: Dr. Dario Rodriguez
Mg. Hernán Amatriain
Lic. Santiago Bianco
Lic. Sebastian Martins

Índice

1. Proceso de Pruebas

Existen distintos tipos de pruebas:

- Pruebas unitarias: El proceso de prueba se centra en los procesos lógicos internos del software, asegurando que todas las sentencias estén probadas y en las funciones externas, buscando que la entrada definida produzca resultados reales de acuerdo con los resultados requeridos. Las pruebas de unidades aseguran que los programas de aplicaciones funcionen de forma adecuada cuando se prueban de forma aislada con respecto a otros programas de aplicación.
- Pruebas de integración: se buscan errores en la interrelación entre los diferentes componentes del sistema, o relación integral entre los subsistemas
- Pruebas del sistema: Las pruebas de sistemas aseguran que los programas de aplicaciones escritos de forma aislada funcionen adecuadamente cuando se integran en el sistema global. La entrada es la especificación de requerimientos del sistema.
- Pruebas que se realizan en la implementación: Pruebas de usuarios, pruebas de control de calidad, pruebas piloto.

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. En cualquiera de los tipos de pruebas es posible considerar distintos aspectos a probar, como ser: la funcionalidad del programa, el volumen de transacciones y la concurrencia de las transacciones.

Las actividades básicas de un procedimiento de pruebas y la documentación de salida correspondiente a cada una de estas actividades son las siguientes:

Actividades	Salidas
Planificación de la prueba	Plan de pruebas
Diseño de la prueba	Documentación de diseño de prueba
Determinación de los casos de prueba	Especificación de los casos de prueba
Planificación del procedimiento de prueba	Especificación del procedimiento de prueba
Ejecución de la prueba	Informe de los casos de prueba
Análisis y evaluación de la prueba	Informe de la prueba

- Planificación de la prueba: En la documentación para la planificación de la prueba se deberá indicar:
  - Objetivo de la prueba, detectar fallas,
  - Objetos a probar, funcionalidades, procesos, etc.,
  - Características a probar, dónde se probará, bajo qué entorno, etc.,
  - Características a no probar,
  - Método de prueba, el método utilizado en el desarrollo de la prueba,
  - Tamaño de la prueba, la cantidad de casos de prueba a realizar,
  - Recursos tecnológicos o máquinas. Recursos humanos, mínimo óptimo 3 personas asignadas, un planificador, un diseñador y un testeador,
  - Reparto de responsabilidades, qué tareas tiene asignadas cada integrante,
  - Plan de tiempo, Diagrama de Gantt,
  - Productos a generar en el proceso de prueba, toda la documentación que tendrá la prueba una vez finalizada.

- Diseño de la prueba: En el diseño se deberá puntualizar con:
  - Cómo llevar adelante la prueba para alcanzar el objetivo establecido,
  - De qué forma se va a usar el método de prueba, brevemente qué método se utilizará y qué probará ese método,
  - Criterios para pasar la prueba, es el punto más importante e indica qué escala utilizaremos para determinar si algo pasa la prueba o no.

- Determinación de los casos de prueba: Determinar puntualmente los casos de prueba, es decir, enumerar uno a uno los casos de pruebas a probar, indicar el resultado correcto esperado al realizar algún caso de prueba, pudiéndose utilizar una tabla de la siguiente forma:

Función a probar (objeto)	Valor a ingresar	Objetivo	Resultado esperado

Básicamente estas cuatro columnas

- Planificación del procedimiento de prueba: Se especificarán los procedimientos que es necesario seguir durante la ejecución de la prueba.

- Ejecución de la prueba: Es en sí la ejecución de la prueba propiamente dicha, de acuerdo a la planificación del procedimiento de prueba.

Función a probar (objeto)	Valor a ingresar	Objetivo <opcional>	Resultado esperado	Resultado obtenido

Básicamente estas cinco columnas

- Análisis y evaluación de la prueba: Se redactarán los informes necesarios sobre el resultado de las pruebas. Cada integrante involucrado puede realizar informes de análisis y evaluación. Se deberá indicar en forma clara y concisa los resultados de la prueba.

A continuación se presenta ejemplo sobre la planificación de pruebas para un sistema que permite realizar un ABMC de una lista de números.

I. *Plan de pruebas*

F420-A	***Hoja 1/3***
00001	**PLAN DE PRUEBA DE PROGRAMA**
10/04 /2009 *Fecha emisión*	***Proyecto:*** ***Ingeniería del Software I*** ***Programa:*** ***Ejemplo - Lista*** ***Autor:*** *RGM*

Objetivo de la prueba:
Determinar las fallas en el procedimiento del programa Ejemplo –Lista

Objetos a probar:

1. *Abrir fichero*
2. *Leer fichero*
3. *Añadir "n" elementos*
4. *Añadir un elemento*
5. *Modificar elemento*
6. *Borrar elemento*
7. *Salir del sistema*

Características a probar:
Funcionalidad de cada uno de los objetos a probar, en una plataforma Windows Vista, con base de datos Access.

Características a no probar:

- *No se tendrá en cuenta otra plataforma que no sea Windows 95.*
- *Velocidad en las operaciones realizadas por el sistema*

F420-A 00001 10/04/2009 Fecha emisión	PLAN DE PRUEBA DE PROGRAMA Proyecto: Ingeniería del Software I Programa: Ejemplo - Lista	Hoja 2/3 Autor: RGM

Cantidad de casos de prueba: 14

Método de prueba a utilizar: Adivinación de errores

Recursos a utilizar:

Técnicos:

- PC Celeron, con 80 GB, 2 Gb de memoria RAM.
- Lenguaje de programación: Visual Basic
- Base de datos: Access
- Procesador de texto: Word 2007
- Impresora: Hewlett Packard Multifunción

Humanos:

- Jefe de proyecto, RGM: 16 horas.
- Asistente, encargado de realizar las pruebas, PB: 4 horas.
- Un programador, DR: 8 horas.

Plan de tiempos:

Planificación de la prueba_____ 4 horas.
Diseño de las pruebas_____ 4 horas.
Ejecución de la prueba_____ 2 horas.
Evaluación de la prueba_____ 2 horas.
Codificación (de ser necesario)_____ 8 horas.
TOTAL_____ 20 horas

F420-A 00001 10/04/2009 Fecha emisión	Hoja 3/3 **PLAN DE PRUEBA DE PROGRAMA** **Proyecto:** *Ingeniería del Software I* **Programa:** *Ejemplo - Lista* **Autor:** *RGM*

Productos a generar durante el proceso de pruebas:

- *Plan de pruebas,*
- *Documento de diseño de pruebas,*
- *Especificación de los casos de prueba,*
- *Especificación de los procedimientos de prueba,*
- *Informe de los casos de pruebas ejecutados,*
- *Informe de pruebas,*
- *Anexo con listados de las pruebas realizadas.*

Reparto de responsabilidades:

Planificación de la prueba: RGM
Diseño de las pruebas: RGM
Ejecución de la prueba: PB
Evaluación de la prueba: RGM, PB
Codificación (de ser necesario): DR

Observaciones:

Jefe de Proyecto

V º B º *15 / 04/ 2009*

Firma

II. *Documento de diseño de la prueba*

F420-B	**DISEÑO DE LA PRUEBA**	
00001	***Proyecto:*** ***Ingeniería del Software I***	
16/04 / 2009 *Fecha emisión*	***Programa:*** ***Ejemplo - Lista***	***Autor:*** *RGM*

Procedimiento de pruebas:
Las pruebas serán llevadas a cabo de acuerdo a lo descrito en el documento *F420-D. Especificación de los casos de prueba,* registrándose las anomalías planteadas.

No se deberá realizar ningún tipo de cambio, sin la previa autorización por escrito del jefe de proyecto.

Métodos de prueba a utilizar:
Se utilizara el método de caja negra, adivinación de errores, para poder así determinar las posibles fallas del sistema en cuanto a la funcionalidad

Criterios para aprobación de pruebas:
Los criterios para la aprobación de las pruebas se realizaran de acuerdo con la siguiente tabla:

- ***Excelente:*** *Cuando el resultado obtenido luego de realizada la prueba es idéntico al resultado citado en la Especificación de pruebas.*
- ***Muy Bueno:*** *Cuando el resultado obtenido luego de realizada la prueba es parecido al resultado citado en la Especificación de pruebas.*
- ***Bueno:*** *Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la Especificación de pruebas, pero no ha provocado anomalías en el funcionamiento del programa.*
- ***Regular:*** *Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la Especificación de pruebas, y ha provocado anomalías en el funcionamiento del programa.*
- ***Malo:*** *Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la Especificación de pruebas, y ha provocado anomalías en el funcionamiento del programa tales como la salida del sistema o "colgarse"*

Jefe de Proyecto

V° B°

16/04/2009

Firma

III. Especificación de los casos de prueba:

F420-C 00001 17/ 04/2009 Fecha emisión	ESPECIFICACIÓN DE LOS CASOS DE PRUEBA **Proyecto:** Ingeniería del Software I **Programa:** Ejemplo - Lista **Autor:** RGM

Especificación de los casos de pruebas:

Ítem	Objetivo	Acción	Entrada	Resultado Esperado
1	Abrir fichero inexistente	Ingresar a la opción 1: Abrir fichero	A:\Lista.txt	Mensaje de error: "Archivo Inexistente"
2	Abrir fichero existente	Ingresar a la opción 1: Abrir fichero	C:\Lista.txt	
3	Leer lista vacía	Ingresar a la opción 2: Leer fichero		Mensaje de error: "Lista vacía"
4	Añadir "n" elementos	Ingresar a la opción 3: Añadir "n" elementos - Cantidad de elementos:	3	
5	Añadir elementos	Ingresar a la opción 3: Añadir "n" elementos – Elementos:	5.3;1.3;6.2	n1=5.3; n2=1.3; n3=6,2
6	Modificar un elemento	Ingresar a la opción 5: Modificar elemento – Nro. elemento	4	Mensaje de error: "Elemento inexistente"
7	Modificar un elemento	Ingresar a la opción 5: Modificar elemento – Nro. elemento	3	
8	Modificar un elemento	Ingresar a la opción 5: Modificar elemento – Nuevo valor :	2	n3=2
9	Leer lista modificada	Ingresar a la opción 2: Leer fichero		n1=5.3; n2=1.3; n3=2
10	Añadir un elemento	Ingresar a la opción 4: Añadir un elemento	5.2	n1=5,2
11	Leer lista modificada	Ingresar a la opción 2: Leer fichero		n1=5.3; n2=1.3; n3=2; n4=5.2
12	Borrar un elemento	Ingresar a la opción 6: Borrar un elemento - Nro. de elemento a borrar		Mensaje: "Elemento borrado"
13	Leer lista modificada	Ingresar a la opción 2: Leer fichero		n1=5.3; n2=2; n3=5.2
14	Salir del programa	Ingresar a la opción 7: Salir		

	Jefe de Proyecto
V° B°	*20/04/2009*
	Firma

IV. *Especificación del procedimiento de prueba:*

F420-D ***00001*** *25/04/2009* *Fecha emisión*	**ESPECIFICACIÓN DEL PROCEDIMIENTO DE PRUEBA** ***Proyecto:*** ***Ingeniería del Software I*** ***Programa:*** ***Ejemplo - Lista*** ***Autor:*** *RGM*

Especificación de los procedimientos de pruebas:

Ítem	*Acción*	*Entrada*	*Resultado Esperado*
1	*Ingresar a la opción 1: Abrir fichero*	*A:\Lista.txt*	*Mensaje de error: "Archivo Inexistente"*
2	*Ingresar a la opción 1: Abrir fichero*	*C:\Lista.txt*	
3	*Ingresar a la opción 2: Leer fichero*		*Mensaje de error: "Lista vacía"*
4	*Ingresar a la opción 3: Añadir "n" elementos - Cantidad de elementos:*	*3*	
5	*Ingresar a la opción 3: Añadir "n" elementos - Elementos:*	*5.3;1.3;6.2*	*n1=5.3; n2=1.3; n3=6,2*
6	*Ingresar a la opción 5: Modificar elemento – Nro. elemento*	*4*	*Mensaje de error: "Elemento inexistente"*
7	*Ingresar a la opción 5: Modificar elemento – Nro. elemento*	*3*	
8	*Ingresar a la opción 5: Modificar elemento - Nuevo valor :*	*2*	*n3=2*
9	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=1.3; n3=2*
10	*Ingresar a la opción 4: Añadir un elemento*	*5.2*	*n1=5,2*
11	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=1.3; n3=2; n4=5.2*
12	*Ingresar a la opción 6: Borrar un elemento – Nro. de elemento a borrar*		*Mensaje: "Elemento borrado"*
13	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=2; n3=5.2*
14	*Ingresar a la opción 7: Salir*		

	Jefe de Proyecto
V° B°	*25/04/2009*
	Firma

V. *Informe de los casos de prueba ejecutados:*

F420-E ***00001*** *26/04 /2009* *Fecha emisión*	**CASOS DE PRUEBA EJECUTADO** ***Proyecto:*** ***Ingeniería del Software I*** ***Programa:*** ***Ejemplo - Lista*** ***Autor:*** *PB*

Casos de prueba:

Ítem	*Acción*	*Entrada*	*Resultado Esperado*	*Resultado Obtenido*
1	*Ingresar a la opción 1: Abrir fichero*	*A:\Lista.txt*	*Mensaje de error: "Archivo Inexistente"*	*Excelente*
2	*Ingresar a la opción 1: Abrir fichero*	*C:\Lista.txt*		*Excelente*
3	*Ingresar a la opción 2: Leer fichero*		*Mensaje de error: "Lista vacía"*	*Bueno*
4	*Ingresar a la opción 3: Añadir "n" elementos – Cantidad de elementos:*	*3*		*Excelente*
5	*Ingresar a la opción 3: Añadir "n" elementos – Elementos:*	*5.3;1.3;6.2*	*n1=5.3; n2=1.3; n3=6,2*	*Excelente*
6	*Ingresar a la opción 5: Modificar elemento – Nro. elemento*	*4*	*Mensaje de error: "Elemento inexistente"*	*Muy Bueno*
7	*Ingresar a la opción 5: Modificar elemento – Nro. elemento*	*3*		*Muy Bueno*
8	*Ingresar a la opción 5: Modificar elemento – Nuevo valor :*	*2*	*n3=2*	*Bueno*
9	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=1.3; n3=2*	*Excelente*
10	*Ingresar a la opción 4: Añadir un elemento*	*5.2*	*n1=5,2*	*Excelente*
11	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=1.3; n3=2; n4=5.2*	*Excelente*

12	*Ingresar a la opción 6: Borrar un elemento - Nro. de elemento a borrar*		*Mensaje: "Elemento borrado"*	*Muy bueno*
13	*Ingresar a la opción 2: Leer fichero*		*n1=5.3; n2=2; n3=5.2*	*Excelente*
14	*Ingresar a la opción 7: Salir*			*Excelente*

Jefe de Proyecto

V° B°

27/04/2009

Firma

VI. *Informe de la prueba:*

F420-F ***00001*** *29/04/2009* *Fecha emisión*	**INFORME DE PRUEBA DE PROGRAMA** ***Proyecto:*** ***Ingeniería del Software I*** ***Programa:*** ***Lista*** ***Autor:*** *RGM-PB*

Comentarios de la prueba:

El programa funcionó correctamente en los casos 1, 2, 4, 5, 9, 10, 11, 13 y 14 de acuerdo a las pruebas realizadas.

Con respecto a los ítems 6, 7 y 12 no han aparecido los mensajes esperados, en estilo de mensajes aparecido es similar a los esperados, se ha podido constatar que las funciones han sido realizadas adecuadamente.

En el caso los ítems 3 y 8 no han aparecido los mensajes esperados, pero se ha podido constatar que las funciones han sido realizadas adecuadamente.

Las interfaces de usuario responden adecuadamente.

Recomendaciones:

Dar por cumplimentada la prueba. Adjuntamos listados que constata las pruebas realizadas	
Responsable de la Prueba *29/04/2009* *Firma*	*Jefe de Proyecto* *V ° B °* *29/04 /2009* *Firma*

2. Métodos de prueba

Los métodos de prueba proporcionan un mecanismo de ayuda para asegurar las pruebas sean completas y para conseguir una mayor probabilidad de descubrir errores en el software.

Principalmente los métodos de prueba se dividen en dos conjuntos, los que se basan en revisar la codificación del software sin importar los datos de entrada o salida, y los que, por el contrario, no les interesa la codificación sino los datos de entrada y salida del software. Las primeras se denominan técnicas de caja blanca y las segundas técnicas de caja negra, ambos tipos de técnicas son complementarias.

Pruebas de

- Caja blanca ✳ Codificación
- Estilo de codificación (Condiciones – Flujo de datos – Bucles)
- Caminos independientes

- Caja negra ✳ Funcionalidad (Entrada-Salida)
- Adivinación de errores
- Análisis de valores límite y clases de equivalencia (Análisis de valores de frontera)

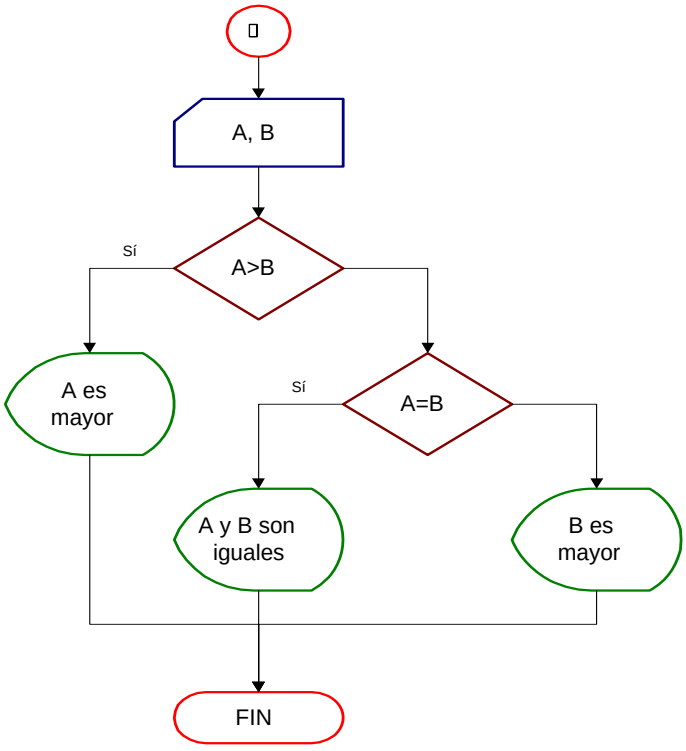
- Estilo de codificación: Se tiene en cuenta los parámetros para una buena codificación. La prueba de condición es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. El método de prueba de flujo de datos selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. La prueba de bucles se centra exclusivamente en la validez de las construcciones de bucles, y éstos pueden ser: simples, concatenados, anidados y no estructurados.
- Adivinación de errores: Se supone qué resultados son los que podrían producir errores.
- Análisis de valores límite (AVL) y clases de equivalencia (CE):
    - Para el AVL se debe seleccionar: 1 valor mayor y 1 valor menor del campo a analizar.
    - Para cada una de las CE se deben elegir:
        - Rangos: 1 caso correcto y 2 casos incorrectos,
        - Lógicos: 1 caso válido y 1 caso inválido, y
        - Textos: 1 caso correcto y 2 casos incorrectos.

A continuación se presenta un ejemplo, dado el siguiente procedimiento:

```
CalcularMayor():
1:   ingresar A, B
2:   if A > B then
3:              mostrar("A es mayor")
          else
4:              if A = B then
5:                        mostrar("A y B son iguales")
                     else
6:                        mostrar("B es mayor")
                endif
7:   endif
```

Se presenta el diagrama de flujo de la situación planteada:

Diagrama de flujo



Teniendo en cuenta que tanto A como B son variables numéricas y tienen definido un rango de: $1000 \leq A \leq 9999$ y $1000 \leq B \leq 9999$; se eligen los valores AVL y CE para A y B:

Valor A		**Valor B**	
AVL	10000	AVL	10000
	999		999
CE	1234	CE	2345

A B
* $

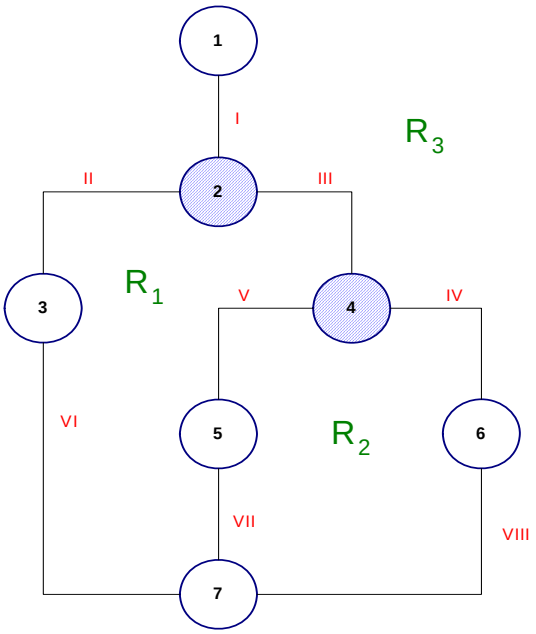
De acuerdo a lo planteado, se sugieren las siguientes pruebas:

Nro.	A	B	Resultado esperado	Resultado obtenido
1	10000	1234	"Mensaje de error"	
2	999	1234	"Mensaje de error"	
3	1234	1234	"A y B son iguales"	
4	A	1234	"Mensaje de error"	
5	*	1234	"Mensaje de error"	
6	1234	10000	"Mensaje de error"	
7	1234	999	"Mensaje de error"	
8	1234	2345	"B es mayor"	
9	1234	B	"Mensaje de error"	
10	1234	$	"Mensaje de error"	

- Caminos independientes: Permite garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo del sistema. Los pasos a seguir son los siguientes:
  - usando el diseño o código como base, se dibuja el correspondiente grafo de flujo,
  - se determina la complejidad ciclomática V(G) del grafo de flujo resultante,
  - se determina un conjunto básico de caminos linealmente independientes,
  - se preparan los casos de prueba que forzarán la ejecución de cada camino del conjunto.

  A continuación se presenta la resolución de este tipo de prueba de acuerdo a la situación planteada:

  1. Usando el diseño o código como base, se dibuja el correspondiente grafo de flujo.



2. Se determina la complejidad ciclomática V(G) del grafo de flujo resultante.

Condiciones ⇒ Nodos predicados 

V(G) = Cantidad de regiones
V(G) = Cantidad de nodos predicado + 1
V(G) = Cantidad de aristas – Cantidad de nodos predicado + 2

V(G) = 3
V(G) = 2 + 1 = 3
V(G) = 8 – 7 + 2 = 3

3. Se determina un conjunto básico de caminos linealmente independientes:
   Camino 1 => 1..2..3..7
   Camino 2 => 1..2..4..5..7
   Camino 3 => 1..2..4..6..7

4. Se preparan los casos de prueba que forzarán la ejecución de cada camino del conjunto.

3. Configuración

En términos muy generales, la Gestión de Configuración del Software (GCS) se puede definir como una disciplina cuya misión es controlar la evolución de un sistema software. La Gestión de Configuración se debe realizar a lo largo de todo el ciclo de vida del producto, tanto en el desarrollo como en el mantenimiento, hasta que el producto se retira.

El objetivo de las actividades de Gestión de Configuración del Software es:

- establecer y mantener la integridad de los productos generados durante un proyecto de desarrollo de software y a lo largo de todo el ciclo de vida del producto.
- evaluar y controlar los cambios sobre ellos, es decir, controlar la evolución del sistema software.
- facilitar la visibilidad sobre el producto.

La integridad de un producto software significa que el producto cumple las siguientes condiciones:

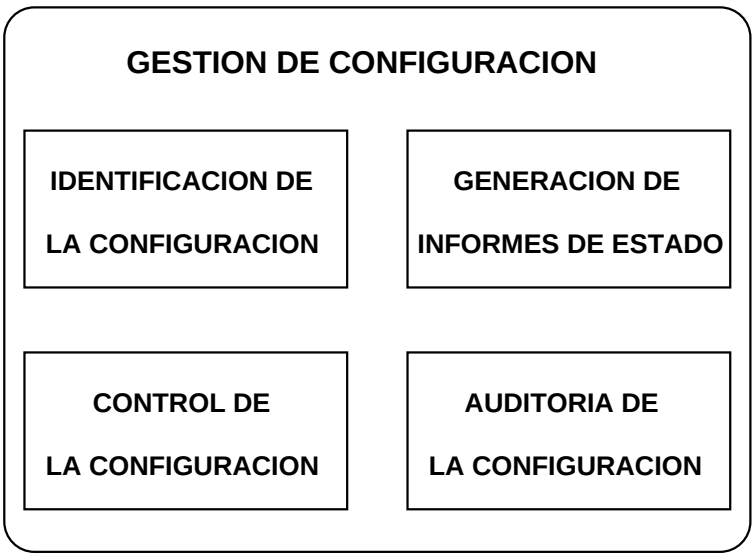
- Satisface las necesidades del usuario (cumple todos los requisitos del usuario, tanto los explícitos como los implícitos)
- Cumple los requisitos de rendimiento
- Se puede trazar su evolución desde que se concibió, y a través de todas las fases de su ciclo de vida.

La definición estándar de Gestión de Configuración del Software, tal y como aparece en el estándar de IEEE, incluye las siguientes actividades:

- Identificación de la Configuración: Consiste en identificar la estructura del producto, sus componentes y el tipo de estos, y en hacerlos únicos y accesibles de alguna forma. Esta tarea consiste en identificar y asignar nombres significativos y consistentes a todos y cada uno de los elementos que forman parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software.
- Control de Cambios en la Configuración: Consiste en controlar las versiones y entregas de un producto y los cambios que se producen en él a lo largo del ciclo de vida.
- Generación de Informes de Estado: Consiste en informar acerca del estado de los componentes de un producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto. El objetivo es mantener a los usuarios, a los gestores y a los desarrolladores al tanto del estado de la

configuración y su evolución. En definitiva, pretende dar respuesta a la pregunta "¿Qué ocurrió?", y también a la pregunta "¿Cuándo ocurrió?".

- Auditoría de la Configuración: Consiste en validar la completitud de un producto y la consistencia entre sus componentes, asegurando que el producto es lo que el usuario quiere.



Sin embargo, los sistemas que automatizan la gestión de configuración suelen incluir algunas funciones adicionales:

- Construcción: Consiste en gestionar la compilación y enlazado de los distintos componentes del producto software de una forma lo más eficiente posible.
- Control del Trabajo en Equipo: Consiste en controlar las interacciones que se producen entre los múltiples desarrolladores de un producto, sobre todo cuando deben compartir ciertos componentes del producto.
- Control de Versiones: Consiste en mantener un registro histórico de las diferentes versiones por las que pasan los componentes de un producto, que permita la recuperación de cualquiera de ellas.
- Gestión de Problemas: Consiste en realizar un seguimiento de la evolución de los problemas que afectan al producto.

Cada vez resulta más evidente que las necesidades de Gestión de Configuración en una organización grande, con aplicaciones software de larga vida, o que requieren del mantenimiento simultáneo de múltiples versiones, son muy grandes, y a veces pueden resultar muy complejas. Sin embargo, la mayor parte de las empresas de desarrollo de software, aún hoy en día, realizan una Gestión de Configuración "bajo mínimos".

¿Por qué es compleja la Gestión de Configuración? Se pueden encontrar varias respuestas a esta pregunta:

Número elevado de componentes a controlar: A medida que progresa el proceso de desarrollo de Software el número de componentes crece rápidamente.

Pero el problema realmente surge porque en el proceso toma parte otra variable: el CAMBIO. Sin importar en qué momento del ciclo de vida del sistema nos encontremos, el sistema informático cambiará, y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida".

Hay que asumir el cambio, no evitarlo. Es importante tener en cuenta que la mayoría de los cambios están justificados, ya que a medida que pasa el tiempo se sabe más acerca del problema y de cómo resolverlo. La Gestión de Configuración está también fuertemente relacionada con el problema del mantenimiento del software. Sin una buena Gestión de Configuración, el mantenimiento de un producto puede ser una verdadera pesadilla.

3.1. Elementos de Configuración

Al conjunto de toda la información y productos utilizados o producidos en un proyecto como resultado del proceso de Ingeniería de Software se le denomina CONFIGURACIÓN DEL SOFTWARE. A cada uno de los componentes de la configuración del software se le va a llamar ELEMENTO DE CONFIGURACIÓN DEL SOFTWARE (ECS). El ECS es la unidad de trabajo para la GCS. Se pueden considerar como Elementos de Configuración del Software los siguientes componentes:

- La especificación del sistema.
- El plan del proyecto software.
- La especificación de requisitos software.
- Un prototipo, ejecutable o en papel.
- El diseño preliminar.
- El diseño detallado.
- El código fuente.
- Programas ejecutables.
- El manual de usuario.
- El manual de operación e instalación.
- El plan de pruebas.
- Los casos de prueba ejecutados y los resultados registrados.
- Los estándares y procedimientos de ingeniería de software utilizados.

- Los informes de problemas.
- Las peticiones de mantenimiento.
- Los productos hardware y software utilizados durante el desarrollo.
- La documentación y manuales de los productos hardware y software utilizados durante el desarrollo.
- Diseños de bases de datos.
- Contenidos de bases de datos.
- etc.

A medida que progresa el proceso de desarrollo, el número de ECS crece rápidamente. La Especificación del Sistema da lugar al Plan del Proyecto y a la Especificación de Requisitos Software. A su vez estos dan lugar a otros documentos, etc. Si simplemente cada ECS produjera otros ECS, no habría prácticamente confusión. El problema aparece cuando entra en juego la variable CAMBIO.

3.2. Líneas base

Como ya hemos visto, uno de los objetivos principales de la Gestión de Configuración va a ser el de gestionar los cambios que se producen en el sistema a lo largo de su ciclo de vida. Para controlar los cambios sin impedir los cambios justificados se utiliza el concepto de LÍNEA BASE o "BASELINE".

Se puede definir una línea base como un punto de referencia en el proceso de desarrollo del software que queda marcado por la aprobación de uno o varios Elementos de Configuración del Software, mediante una revisión técnica formal.

La idea consiste en permitir cambios rápidos e informales sobre un Elemento de Configuración del Software antes de que se pase a formar parte de una línea base, pero en el momento en que se establece una línea base se debe aplicar un procedimiento formal para evaluar y verificar cada cambio.

El concepto clave para realizar esta actividad es el de Línea Base. Las líneas base se establecen en hitos previamente especificados a lo largo del proceso de desarrollo. Aunque se pueden definir las líneas base con cualquier nivel de detalle, las líneas base más comunes son:

- Línea Base Funcional, que se establece al finalizar la fase de análisis y especificación de los requisitos del sistema, y comprende todos aquellos documentos en los que se define el problema a resolver, los costes del

proyecto, el plan de tiempos, y los diferentes requisitos funcionales, de interoperatividad y de interfaz del sistema.

- Línea Base de Distribución o Asignación de funciones, que se establece al finalizar la fase de análisis y especificación de requisitos software, y comprende toda la documentación que gobernará el desarrollo de cada uno de los componentes software que se han identificado en la especificación del sistema, y la asignación o reparto de las diferentes funciones entre los distintos componentes del sistema
- Línea Base de Diseño Preliminar, que se establece al finalizar la fase de diseño preliminar. Comprende todos aquellos documentos en los que se define la arquitectura del producto software, así como el Plan de Pruebas.
- Línea Base de Diseño, que se establece al finalizar la fase de diseño detallado. Comprende todos aquellos documentos que contienen el diseño detallado del software y el plan de implementación, y también la descripción de los casos de prueba.
- Línea Base de Producto, que se establece al finalizar la fase de pruebas. Comprende los programas creados y todos aquellos documentos que contienen la información relativa a las pruebas realizadas.
- Línea Base de Operación, que se establece al finalizar la fase de implantación. Comprende los manuales de usuario, guías de operación y mantenimiento, manuales de formación, etc.

3.3. Control de Cambios en la Configuración

Es la actividad de Gestión de Configuración más importante y su objetivo es proporcionar un mecanismo riguroso para controlar los cambios, partiendo de la base de que los cambios se van a producir. Normalmente combina procedimientos humanos y el uso de herramientas automáticas.

Se pueden considerar fundamentalmente dos tipos de cambios:

- Corrección de un defecto: Los clientes tienden a clasificar todos los cambios en esta categoría.
- Mejora del sistema: Los programadores, sin embargo, los suelen clasificar aquí.

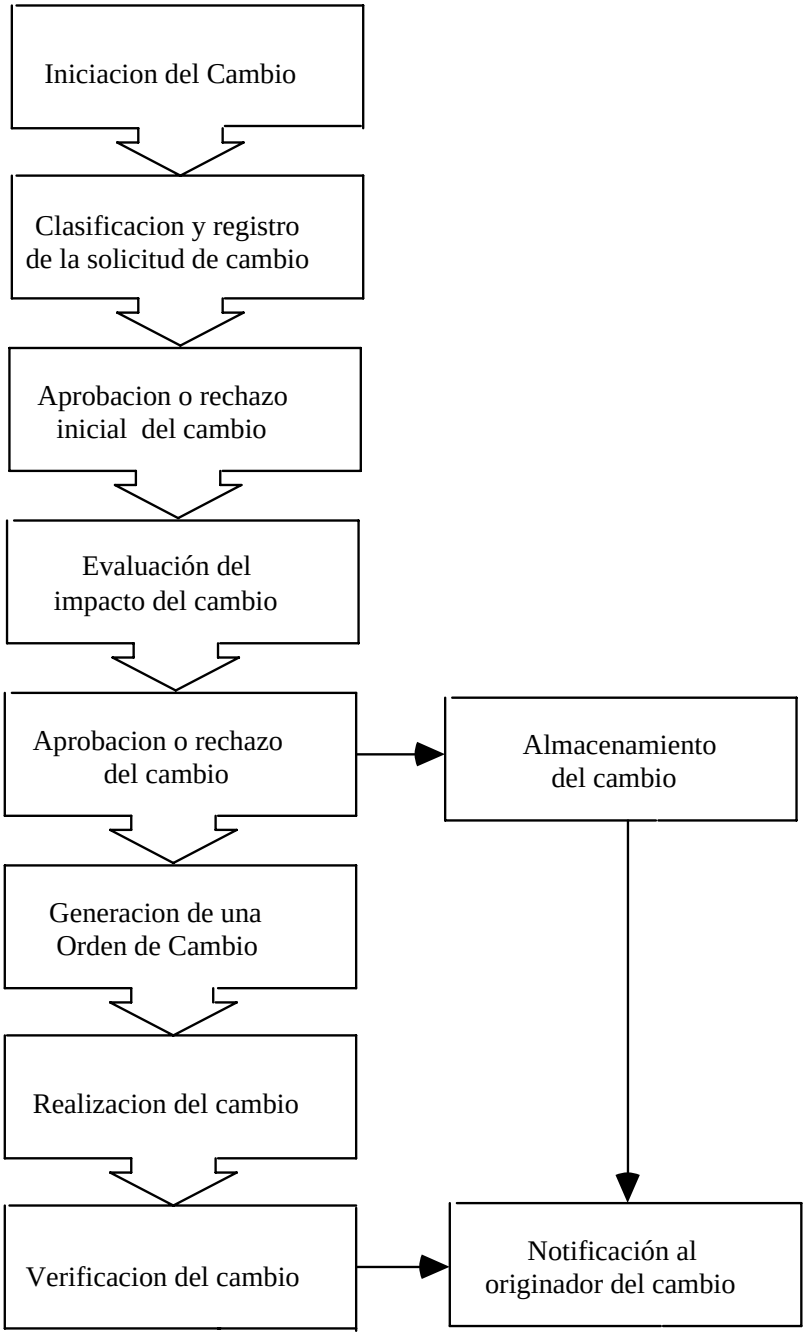
Por lo general se establecen varios niveles de control de cambios:

- Control de cambios informal: Antes de que el Elemento de Configuración del Software pase a formar parte de una línea base, aquel que haya

desarrollado el Elemento de Configuración del Software podrá realizar cualquier cambio justificado sobre él.

- Control de cambios al nivel del proyecto o semi-formal: Una vez que el Elemento de Configuración del Software pasa la revisión técnica formal y se convierte en una línea base, para que el encargado del desarrollo pueda realizar un cambio debe recibir la aprobación de:

    - El director del proyecto, si es un cambio local
    - El Comité de Control de Cambios, si el cambio tiene algún impacto sobre otros
    - Elementos de Configuración del Software
- Control de cambios formal: Se suele adoptar una vez que se empieza a comercializar el producto, cuando se transfieren los ECS a la Biblioteca Maestra. Todo cambio deberá ser aprobado por el Comité de Control de Cambios.



Las etapas típicas de un proceso formal, es decir, el proceso que habría que seguir para hacer un cambio sobre una línea base:

- Iniciación del Cambio: se presenta una solicitud de cambio, que puede venir provocada por un problema que se ha detectado o por un cambio en los requisitos.

- Clasificación y registro de la solicitud de cambio.
- Aprobación o rechazo inicial de la solicitud de cambio. De ello suele ser responsable el Comité de Control de Cambios.
- Evaluación de la solicitud de cambio, si ha sido aprobada, para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio. Como resultado se obtiene un Informe de Cambio.
- Se presenta el Informe de Cambio al Comité de Control de Cambios. Si se considera que el cambio es beneficioso se genera una Orden de Cambio (también llamada Orden de Cambio de Ingeniería), que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría. Esta Orden de Cambio es asignada a alguno de los ingenieros de software para que se encargue de llevarlo a cabo. En este momento, el objeto a cambiar se da de baja en la Biblioteca de Soporte al Proyecto.
- Se realiza el cambio, entrando en un proceso de seguimiento y control.
- Una vez finalizado el cambio, se certifica, mediante una revisión, que se ha efectuado correctamente el cambio y con ello se ha corregido el problema detectado o bien se han satisfecho los requisitos modificados. El objeto se devuelve a la Biblioteca de Soporte al Proyecto.
- Se notifica el resultado al originador del cambio.

Algunos ejemplos de los tipos de registros que pueden mantenerse son los siguientes:

- Registro de elementos de configuración: Conteniendo toda la información relativa a los diferentes elementos de configuración.
- Registro de líneas base: Conteniendo toda la información relativa a cada línea base
- Registro de solicitudes de cambios: El tipo de información que se suele mantener acerca de cada solicitud de cambio es la recogida a través del formulario de Solicitud de Cambio.
- Registro de cambios: El tipo de información que se suele mantener acerca de cada cambio es la recogida a través de el Informe de Cambio, la Orden de Cambio, el proceso de Gestión de Problemas, etc.
- Registro de modificaciones del código
- Registro de modificaciones sobre bases de datos
- Registro de modificaciones sobre documentación
- Registro de instalaciones: Su objetivo es mantener información acerca de todos los lugares en los que se ha instalado un producto software.
- Actas de las reuniones del Comité de Control de Cambios

En cuanto a los informes, podemos distinguir dos tipos:

- Planificados
- Bajo demanda

Algunos ejemplos de los tipos de informes que se pueden generar son:

- Informe de estado de los cambios: Es un resumen del estado en que se encuentran todas las solicitudes de cambio registradas durante un determinado período de tiempo.
- Inventario de elementos de configuración, para ofrecer visibilidad sobre el contenido de las bibliotecas de proyecto.
- Informe de incidencias: Es un resumen del estado en que se encuentran todas las incidencias originadas durante un determinado período de tiempo y las acciones a las que han dado lugar.
- Informe de modificaciones: Es un resumen de las modificaciones que se han efectuado en el producto software durante un determinado período de tiempo.
- Informe de diferencias entre versiones: Resumen de las diferencias entre las sucesivas versiones de un elemento de configuración.

En cualquier caso, al comienzo de cada proyecto será necesario decidir qué tipo de registros se van a mantener y qué tipo de informes se van a generar y para quién.

3.4. Auditoria de la Configuración

Una auditoría es una verificación independiente de un trabajo o del resultado de un trabajo o grupo de trabajos para evaluar su conformidad respecto de especificaciones, estándares, acuerdos contractuales u otros criterios. La auditoría de la Configuración es la forma de comprobar que efectivamente el producto que se está construyendo es lo que pretende ser.

Se pueden diferenciar tres tipos de actividades:

- Revisiones de fase: Se realizan al finalizar cada fase del desarrollo y su objetivo es examinar los productos de dicha fase. Las revisiones propias de la Gestión de configuración son aquellas en las que se establecerán las líneas base. El objetivo de esta revisión es descubrir problemas, no comprobar que todo está bien. Hay que ser capaz de desenmascarar los problemas ocultos y sutiles, no sólo los que son obvios.

- Revisiones de cambios: Se realizan para comprobar que los cambios aprobados sobre una línea base se han realizado correctamente.
- Auditorías: Se realizan al final del proceso de desarrollo de software y su objetivo es examinar el producto en su conjunto.

Las revisiones se deben realizar de forma continua, durante todo el proceso de desarrollo, y no sólo al finalizar éste, cuando los problemas ya no tienen solución.

La tarea de revisión implica tres tipos de funciones:

- Verificar que la configuración actual del software se corresponde con lo que era en fases anteriores. Debe haber correspondencia y trazabilidad entre los elementos de configuración que aparecen en una línea base y los que aparecen en las líneas base que la preceden y que la siguen. La verificación se realiza con respecto a la línea base precedente.
- Validar que la configuración actual del software satisface la función que se esperaba del producto en cada hito del proceso de desarrollo. La validación se realiza con respecto a los requisitos del sistema.
- Valorar si una determinada línea base, teniendo en cuenta los resultados de la verificación y validación, y otro tipo de comprobaciones, se debe considerar aceptable o no.

En cuanto a las auditorías, se suelen distinguir dos tipos de auditorías de configuración:

- Auditoría Funcional: Cuyo objetivo es comprobar que se han completado todos los tests necesarios para el Elemento de Configuración auditado, y que, teniendo en cuenta los resultados obtenidos en los tests, se puede afirmar que el Elemento de Configuración satisface los requisitos que se impusieron sobre él.
- Auditoría Física: Cuyo objetivo es verificar la adecuación, completitud y precisión de la documentación que constituye las líneas base de diseño y de producto. Se trata de asegurar que representa el software que se ha codificado y probado. Tras la auditoría física se establece la línea base de Producto. Tiene lugar inmediatamente después de haberse superado la auditoría Funcional.

Y aún se puede considerar un tercer tipo de auditoría:

- Revisión Formal de Certificación: Cuyo objetivo es certificar que el Elemento de Configuración del Software se comporta correctamente una vez que éste se encuentra en su entorno operativo.

4. Auditoria

La auditoría en informática es la revisión y evaluación de los controles, sistemas, procedimientos de informática y de los equipos de cómputo, su utilización, eficiencia y seguridad, a fin de que por medio del señalamiento de cursos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones.

La auditoría en informática deberá comprender no sólo la evaluación de los equipos de cómputo o de un sistema o procedimiento específico, sino que además habrá de evaluar los sistemas de información en general desde sus entradas, procedimientos, controles, archivos, seguridad y obtención de información. Ello debe incluir los equipos de cómputo como la herramienta que permite obtener la información adecuada y la organización específica (departamento de cómputo, departamento de informática, etc.) que hará posible el uso de los equipos de cómputo.

La auditoría informática es un examen, pues debe partir de una situación dada; éste es metódico, puesto que seguirá un plan de trabajo perfectamente sistematizado que permite llegar a conclusiones suficientemente justificadas (está es una conclusión exigible a cualquier auditoría); es puntual ya que se da un corte en el calendario para llevarla a cabo, y es extraña al servicio de informática, para obtener la objetividad requerida, por lo que será ejecutada por personas ajenas al departamento independientes de las funciones a auditar.

El examen de una auditoría informática abarca una serie de controles, verificaciones, juicios, etc. para concluir en un conjunto de recomendaciones y un plan de acción. Es la elaboración de este plan de acción lo que diferencia la auditoría informática de una auditoría de gestión. La auditoría tradicional concluye emitiendo un juicio del estado de todo aquello que se ha verificado, la auditoría informática avanza un paso más y se atreve a elaborar un plan de acción.

Como se ve la evaluación a desarrollar para la realización de la auditoría en informática deben hacerla personas con alto grado de conocimiento en informática y con mucha experiencia en el área.

5. Trabajo Práctico

De acuerdo al siguiente enunciado:
Se necesita un programa que permita el ingreso de 3 valores enteros (A, B y C) y se determine que tipo de triangulo forma. Recuerde que una propiedad para formar un triangulo es que la suma de dos lados es mejor que el tercer lado.

Para la situación planteada realice:

1. Procedimiento de pruebas que incluya:
    a. Pruebas de caja negra (análisis de valores limites y partición de equivalencia), y
    b. Pruebas de caja blanca (complejidad ciclomática).
2. Procedimiento de gestión de configuración, simulando el cambio de ingreso de números enteros a continuos.
   NOTA: SOLO SE DEBE LLEVAR ADELANTE LA GESTION DE CONFIGURACION, NO REALIZAR LOS PRODUCTOS INVOLUCRADOS EN LA MISMA.