

## Solución Examen Ingeniería del Software II. 7 de Junio de 2004.

Nombre:

Grupo:

**NOTA: Contesta en hojas separadas a cada pregunta.**

### **Ejercicio 1 (1 punto)**

Contesta Verdadero o Falso a las siguientes preguntas, teniendo en cuenta que los errores descuentan 0,1 puntos (contesta en esta hoja y entrégala junto con tu examen):

1. Las auditorías suelen utilizarse como medio para la certificación de sistemas de calidad frente a normativas estándar. VERDADERO
2. En un proyecto pequeño podemos prescindir de las actividades de gestión del proceso de ingeniería. FALSO.
3. Al hacer mantenimiento correctivo, siempre es necesario modificar el documento de especificación de requisitos. FALSO.
4. Los elementos de una línea base se han revisado y aprobado, sirven como base para actividades posteriores y por tanto no se pueden modificar. FALSO.
5. Todas las actividades posibles que mejoran la mantenibilidad del software se llevan a cabo a lo largo del ciclo de vida de desarrollo de un sistema software. FALSO.
6. El director de un proyecto informático es el encargado de realizar las actividades financieras del proyecto. FALSO.
7. El control de cambios es una actividad protectora que se comienza una vez que se entrega la aplicación. FALSO.
8. Las pruebas de aceptación incluyen casos de caja blanca. FALSO.
9. La gestión de riesgos es una actividad estática, esto es, se realiza una sola vez, al principio de proyecto. FALSO.
10. Las inspecciones suelen ser más eficientes que los walkthroughs (en términos de defectos encontrados), aunque suelen ser más costosas. VERDADERO.

## **Ejercicio 2 (3 puntos)**

Responde a las siguientes preguntas (tienes dos caras como máximo):

a) ¿Qué diferencias hay entre las pruebas de integración de un sistema estructurado y uno orientado a objetos? ¿Y entre las pruebas de unidad?

El diseño de un sistema estructurado tiene estructura jerárquica, y por tanto se puede integrar empezando por arriba en la jerarquía, por abajo o en "sandwich". Esto no sucede con los sistemas orientados a objetos, en los que la integración se puede hacer por ejemplo cogiendo los escenarios de los diagramas de secuencia.

Las pruebas de unidad de los sistemas estructurados consisten en probar las funciones por separado. En los sistemas orientados a objetos, además de probar los métodos, se extienden para probar objetos enteros. Como los objetos tienen estado, se suelen hacer siguiendo los caminos de los Diagramas de Estado asociados.

b) ¿Cuáles son las diferencias y semejanzas entre los frameworks y los patrones de diseño?

Los frameworks son porciones de diseño que se reutilizan añadiendo clases o especializando clases, completando métodos y funciones callback, etc. Suelen implementar una parte importante de la funcionalidad (por ejemplo, la interfaz de usuario). La reutilización de un framework es a nivel de diseño y código. Un ejemplo podrían ser las MFCs, o el AWT de Java. Los patrones de diseño son a nivel de diseño y a menor escala, ya que son soluciones "estándar" (probadas y buenas) a problemas comunes, pero más reducidos.

c) ¿Cuáles son las diferencias y semejanzas, en cuanto a la manera de realizarse y los objetivos, entre las auditorías y las inspecciones?

Ambas son medidas analíticas estáticas de aseguramiento de calidad. Las auditorías se suelen realizar al final de proyecto (en el caso de que se hagan), y son investigaciones sobre el producto o el proceso, para determinar lo que fue bien y mal. Incluyen reuniones con los participantes del proyecto, estudio de documentos, observaciones, etc. También son un medio para determinar el grado de cumplimiento de los procesos de una empresa a un estándar (revisiones de certificación). Las inspecciones son actividades técnicas de verificación, en las que varias personas con un rol bien definido (moderador, lector, etc.) analizan un producto intermedio para encontrar defectos.

d) En el contexto de la Ingeniería del Software ¿Qué son las métricas? ¿Para qué valen?

Son mediciones (o combinaciones de ellas), que pueden ser del producto, proceso o proyecto. Sirven por ejemplo para predecir (estimar) parámetros de futuros proyectos, controlar y evaluar el estado del proyecto (comparando con los valores predichos) o la calidad del producto y procesos. Son la base de la mejora de procesos.

e) ¿Cuándo es necesaria la ingeniería inversa? ¿Qué tipos de ingeniería inversa existen? ¿Cuáles son sus entradas y salidas?

Cuando queremos recuperar un nivel de abstracción mayor al que tenemos, típicamente el diseño a partir del código. Es útil por ejemplo en la fase de mantenimiento, cuando no tenemos documentación

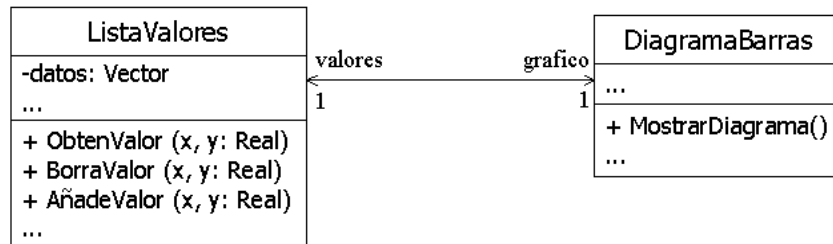
de diseño o está desactualizada. Existen dos tipos: de datos y lógica. Del primer tipo, la entrada son los datos de la aplicación (BBDD), y la salida por ejemplo, diagramas entidad relación. Del segundo tipo, la entrada son los programas, y la salida los diagramas de diseño (p.ej.: DEC's si es estructurado, diagramas de clases y de comportamiento en el caso de orientación a objetos).

f) ¿En qué consisten las actividades de seguimiento que hace el jefe de proyecto? ¿Qué tipo de documentos ayudan a la realización de estas actividades?

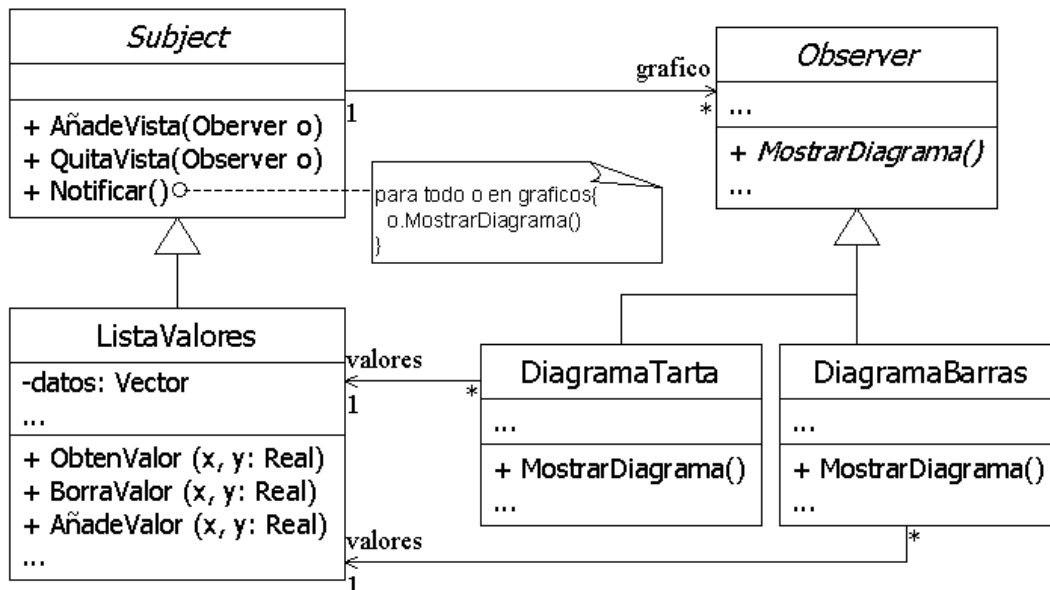
Consisten en determinar el estado (calendario y resultados) del proyecto (tomando métricas, realizando reuniones, etc.) en ciertos momentos, y comparar con la planificación. Se modifica la planificación (y se toman acciones correctivas) en caso de que los datos reales no concuerden con el plan. También debe hacerse un seguimiento de los riesgos identificados, y emplear los planes de contingencia en caso de que los riesgos se presenten. La ayuda principal son los planes (de proyecto, de gestión de riesgos, etc.), porque ofrecen una guía al jefe de proyecto de cómo debería desarrollarse el proyecto, qué actividades debería realizarse, etc.

### Ejercicio 3 (1 punto)

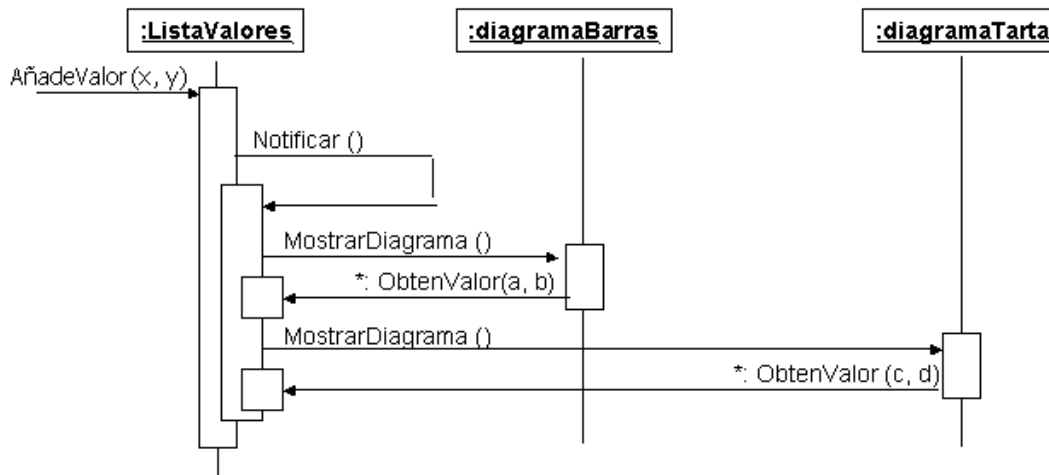
El siguiente diagrama de clases representa parte del diseño de una aplicación estadística. En particular, es una parte de la interfaz de usuario, que representa mediante un diagrama de barras (clase DiagramaBarras) una serie de datos (pares X e Y, implementados en la clase ListaValores). El cliente te pide la inclusión de un nuevo tipo de Diagrama que represente los datos en forma de “tarta”. Modifica el diseño para incluir este cambio, utilizando para ello los patrones de diseño que consideres necesarios. Si lo necesitas, puedes hacer uso de otros diagramas UML para explicar el funcionamiento de tu diseño.



Empleamos el patrón **Observer**, ya que los diagramas son “observadores” de los datos contenidos en *ListaValores*.



El funcionamiento de la solución propuesta se muestra en el siguiente diagrama de secuencia. En él se puede ver que ante una modificación de los datos del sujeto, éste llama al método “Notificar”, que avisa a todos los observadores registrados de que ha habido un cambio. Estos posteriormente llaman a la función “ObtenValor” para actualizar sus datos y mostrarlos adecuadamente en el formato correcto.



### **Ejercicio 4 (2 puntos)**

Dadas las siguientes situaciones, responde brevemente a las preguntas que se formulan:

- a) En un proyecto, al terminar la fase de pruebas, te das cuenta de que están incompletas. Además, faltan importantes detalles de implementación en el documento de diseño. ¿Qué medidas de aseguramiento de calidad aplicarías para solucionar estos problemas en el proyecto actual o prevenirlos en próximos proyectos? (0.5 puntos)

Habría que hacer medidas analíticas dinámicas (realizar las pruebas que faltan). Probablemente en el futuro haya que realizar inspecciones del plan de pruebas para que esto no suceda, y del documento de diseño.

- b) En la empresa “MANUAL S.A.”, se hace excesivo énfasis en la codificación, que además se realiza de manera enteramente manual; hay una ausencia casi total de especificaciones y de procedimientos de aseguramiento de calidad; el mantenimiento de las aplicaciones directamente sobre el código y hay descoordinación entre documentación y situación real de la aplicación. ¿Hasta qué punto crees que la introducción de una herramienta, o una serie de herramientas CASE, solucionaría estos problemas? ¿Qué tipo de herramientas CASE?

1. El problema 1 podría solucionarse con herramientas CASE de análisis y diseño, idealmente con capacidades de generación de código.
2. El problema 2 es un problema que no puede solucionarse con herramientas CASE, sino que es un problema de la (falta de) técnicas apropiadas para la educación y representación de requisitos.
3. En el problema de ausencia de procedimientos de calidad no pueden ayudar, ya que los procedimientos de aseguramiento de calidad han de construirse por el personal de la empresa, describiendo las diversas actividades. Las herramientas pueden facilitar actividades de control de calidad, como las métricas, pruebas dinámicas y análisis estático.
4. El problema de mantenimiento a nivel de código se podría solucionar también con CASE, ya que si a partir de los modelos podemos generar parte o todo el código, podemos dar mantenimiento a nivel de modelos.

5. El problema de la coordinación de documentación y código se podría resolver, porque las herramientas pueden generar parte de la documentación, siendo más fácil mantenerla coordinada cada vez que se hace un cambio.

- c) Eres el jefe de proyecto de una aplicación que está en fase de mantenimiento. La aplicación fue construida en COBOL y corre en el sistema operativo VMS en *mainframe*. La aplicación falla frecuentemente, se quiere portar a una arquitectura más moderna (basada en cliente/servidor) y se quiere añadir nueva funcionalidad. ¿qué tipo de actividades de reingeniería realizarías para mejorar el mantenimiento? (0.5 puntos).

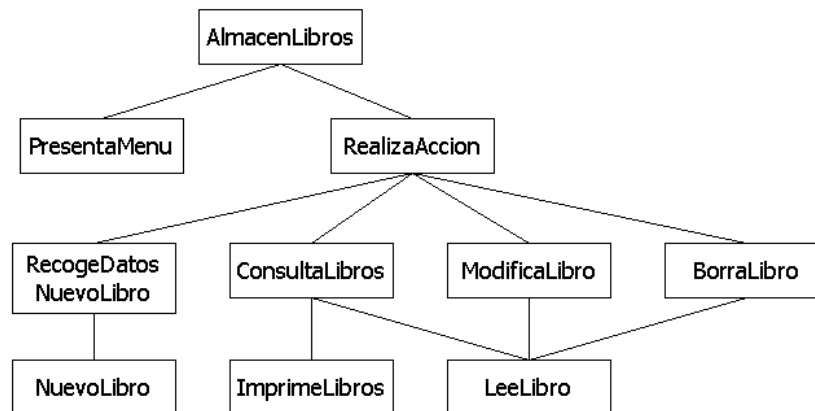
Como hay que cambiar la funcionalidad, y hay problemas de fiabilidad, se podría optar por construir el sistema de nuevo (utilizando el sistema antiguo como prototipo).

- d) En un proyecto temes que algunos miembros clave del equipo de desarrollo dejen la empresa. Como jefe de proyecto, ¿qué puedes hacer para reducir este riesgo?

Para reducir la probabilidad de realización del riesgo, habría que motivarlos y hacer que "estén contentos". Para reducir el impacto, habría que hacer que documenten de manera adecuada y reuniones frecuentes, para que la gente esté enterada de lo que hacen.

### **Ejercicio 5 (3 puntos)**

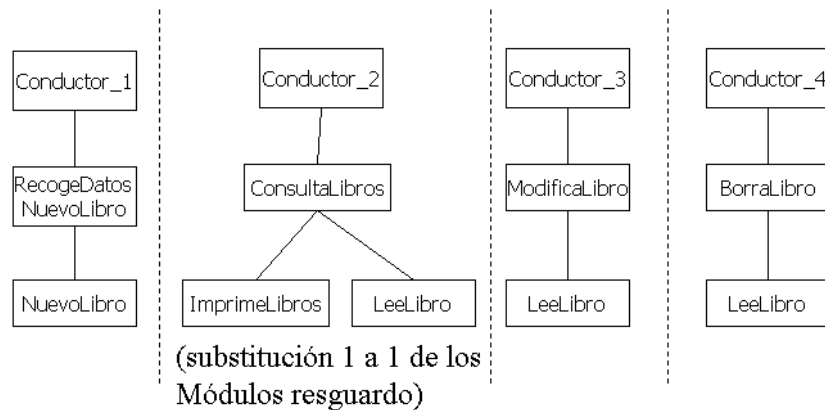
El siguiente diagrama de estructura de cuadros representa una aplicación simple para el manejo de almacén de una tienda de libros.



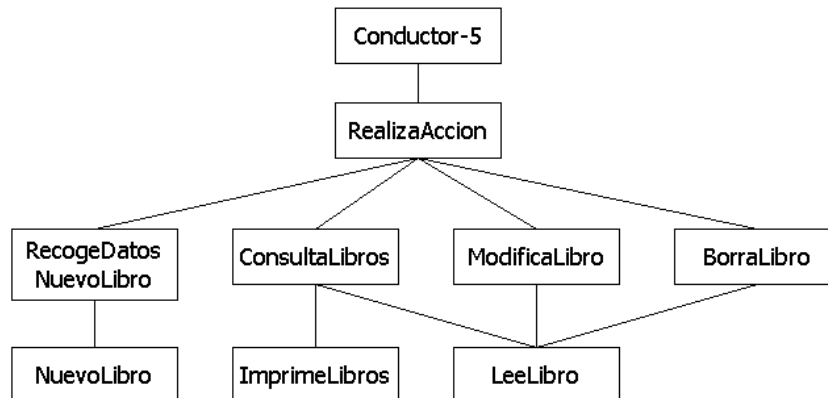
- a) Indica cuáles serían los pasos de integración, suponiendo que no se han realizado las pruebas de unidad y que eliges una estrategia ascendente (muestra DEC's con las agrupaciones que pruebes en cada paso).

Paso 1: Unidad de "NuevoLibro", "ImprimeLibros", "LeeLibro", "RecogeDatosNuevoLibro", "ConsultaLibros", "ModificaLibro", "BorraLibro" y "PresentaMenu" con sus respectivos módulos conductores y resguardos.

Paso 2: Se prueban las siguientes agrupaciones, junto con las pruebas de unidad de "RealizaAccion" con sus respectivos módulos conductores y resguardos.:



Paso 3: Se prueban las siguientes agrupaciones, junto con las pruebas de unidad de "AlmacenLibros" con sus respectivos módulos resguardos:



Paso 4: Todo junto.

- b) Dada la especificación del módulo "NuevoLibro", indica cómo obtendrías un juego de casos de prueba de caja negra. Muestra 10 casos de prueba.

```
#define OK 0
#define ERROR_CONEXION_BD -1
#define ERROR_ID -2
#define ERROR_IDREPETIDO -3
#define ERROR_NOMBRELIBRO -4
#define ERROR_AUTOR -5
#define ERROR_EDITORIAL -6
#define ERROR_PAGINAS -7
#define ERROR_PRECIO -8
#define ERROR_DESCUENTO -9

int NuevoLibro ( char * id, // Identificador del libro, debe ser único y
                // tiene el formato 'LIB' seguido de siete
                // dígitos.
                char * nombreLibro, // Nombre del libro, cadena de 1 a 512
                // caracteres
                char * Autor, // Autor del libro, cadena de 1 a 256
                // caracteres
                char * editorial, // Editorial del libro, cadena de 1 a 128
                // caracteres
                int paginas, // páginas del libro, entero de 1 a 10000
                int anyo, // Año de edición: de 1920 al año actual.
                float precio, // precio, de 0,5 a 200.
                float descuento) // descuento, de 0 a 100%.
```

/\* Se conecta a la base de datos e inserta un nuevo registro, chequeando previamente la consistencia de cada uno de los campos, devuelve 0 en caso de éxito o alguno de los códigos de error definidos previamente.

La nueva normativa de la empresa establece un descuento del 10% para los libros editados entre 1950 y 1970, y un 5% para los libros de 1971 a 1990. Hay un 5% adicional para los libros de la editorial "PLANETA".

\*/

### SOLUCION:

Se utilizarían técnicas de partición de equivalencia, análisis del valor límite y conjetura de error.

Las clases de equivalencia de los parámetros de entrada serían:

<i>Dato</i>	<i>Clase Válida</i>	<i>Clase Inválida</i>
Id	"LIBDDDDDDDD"	3 primeros caracteres <> LIB
		7 caracteres siguientes no dígitos
	Longitud=10	Longitud < 10
		Longitud > 10
nombreLibro	Longitud en [1, 512]	Longitud < 1
		Longitud > 512
Autor	Longitud en [1, 256]	Longitud < 1
		Longitud > 256
Editorial	Longitud en [1, 128]	Longitud < 1
		Longitud > 128
Paginas	Entero en [1, 10000]	Longitud < 1
		Longitud > 10000
Anyo	Entero en [1920, actual]	Anyo < 1920
		Anyo > actual
Precio	Real en [0.5 , 200]	Precio < 0.5
		Precio > 200
Descuento (formato)	Real en [0,100]	Descuento < 0
		Descuento > 100
Descuento (valor)	10% si anyo en [1950, 1970]	Otro valor
	5% si anyo en [1971,1990]	
	+5% si editorial=="PLANETA"	
Conectividad BD	SI	NO
Salida	0	-1 a -9

### 10 CASOS DE PRUEBA:

Se probarían con los valores límite de las clases de equivalencia, combinando los valores de los parámetros, de tal manera que en cada caso de error hay como máximo una clase inválida. También se incluirían casos de conjetura de error (valores típicos de error, punteros a NULL, 0, etc.). En cada caso de prueba habría que incluir el resultado esperado.