Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
Asignatura: Ingeniería del Software I
19 de diciembre de 2007

Evaluación Final

Parte teórica

Ej. 1. Además de la calidad, la Ingeniería del Software debe confrontarse con otros desafíos. Enumérelos y explíquelos brevemente.

Ej. 2.
(a) Además del *proceso de desarrollo*, ¿qué otros procesos conoce dentro del *proceso de software*? Explique brevemente cuál es el fin de cada uno de ellos.
(b) Si usted se da cuenta que su cliente no tiene en claro lo que espera del software solicitado ¿qué modelo de proceso de desarrollo utilizaría? ¿Por qué? ¿Qué desventajas debe confrontar al usar dicho modelo?

Si además el cliente introduce cambios de requerimientos a lo largo del proceso, ¿utilizaría el mismo modelo o utilizaría otro? ¿Por qué?

Ej. 3.
(a) ¿Qué especifican los *requerimientos de funcionalidad*?
(b) ¿Qué otras componentes conforman la especificación de los requerimientos del software? Dé dos ejemplos de características que especifiquen cada una de ellas.

Ej. 4.
(a) ¿Qué describen las vistas de módulo y la de asignación de recursos?
(b) Un sistema cuenta con una base de datos accedida por diversos procesos que responden a accesos a través de la web y que son accedidos remotamente por distintos usuarios. Si usted debe realizar la vista de C&C para un sistema con estas características ¿qué modelos arquitectónicos utilizaría? ¿Por qué?

Ej. 5.
(a) ¿Qué estructuras de equipos de trabajo conoce? Descríbalas muy brevemente.
(b) ¿Cuál es el objetivo de la administración del riesgo? ¿Qué pasos involucra y en qué etapas del desarrollo del proyecto se encuentran?

Ej. 6.
(a) ¿Cuás es el objetivo de la *abstracción*? ¿Qué tipos de abstracción conoce? Descríbalos muy brevemente.
(b) ¿Cuándo se dice que un sistema es *modular*?
(c) ¿Qué es *acoplamiento*, qué es *cohesión*, y qué se espera que un diseño satisfaga respecto de ellos y por qué?

Ej. 7.
(a) Describa el tipo de acoplamiento de *componentes*.
(b) Explique que establece el *principio abierto-cerrado* y cuál es su fin.

Ej. 8. ¿Cuál es el objetivo de la *programación estructurada*? Explíquelo

Ej. 9.

(a) Defina testing de caja blanca y de caja negra.

(b) Dé un pequeño ejemplo de programa erróneo que pase un test por particionado por clases de equivalencias y ***no*** pase un test por anáisis de valores límites. Justifique.

(c) Dé un pequeño ejemplo de programa erróneo para el cual haya un test que cumpla el el criterio de cobertura de nodos que detecte el error y un test que cumpla el el criterio de cobertura de bifurcación que ***no*** lo detecte. Justifique.

Parte práctica

Ej. 11. Considere el siguiente problema y produzca un DFD correspondiente al diseño estructurado para las funcionalidades asociadas a la descripción:

> Se trata de diseñar un sistema que se utilizará para controlar una máquina recicladora de botellas, frascos y cajas. Esta máquina puede ser utilizada por diferentes clientes, y cada cliente puede retornar los tres tipos de artículos (botella, frasco y caja) en la misma ocasión. Dado que hay diferentes tipos de artículos, el sistema debe comprobar, para cada artículo, de que tipo es el devuelto.
>
> También se almacenará cuantos artículos ha devuelto cada cliente. Cuando un cliente solicite un recibo, el sistema imprimirá la cantidad que ha depositado, el valor de esos artículos y el total que se le abonará al cliente.
>
> El sistema también será usado por un operador. Este querrá saber cuantos artículos de cada tipo se han retornado durante el día; esta información se solicitará y se imprimirá al final de cada día. El operador también podrá modificar información dentro de la máquina, como por ejemplo los valores de los depósitos de los diferentes artículos.
>
> Si surge algún problema, como el desborde del recipiente de algún artículo o bien se termina el papel de impresión, el operador será avisado mediante una alarma

Ej. 12. Considere el siguiente problema, y produzca un diagrama de clases correspondiente al análisis orientado a objetos de una solución para el mismo:

> Se desea desarrollar un sistema informático para controlar los préstamos de la biblioteca de una institución. Los estudiantes son los que toman los libros de las estanterías, los leen, los sacan en préstamo, los devuelven y consultan el catálogo de ejemplares disponibles. Cada libro tiene un código de barras que lo identifica. Adicionalmente dispone de un dispositivo electrónico antirrobo situado debajo del código de barras, siendo el sistema el encargado de activarlo y desactivarlo. Cuando un usuario desea obtener en préstamo un libro se sitúa con él delante del sistema de préstamo. Primeramente pasa por el lector su tarjeta de socio. A continuación se produce la lectura del código de barras de los libros que desee obtener en préstamo. Si el socio no tiene préstamos vencidos el sistema desbloquea el código antirrobo del libro. En una misma operación se pueden obtener varios libros en préstamo. El proceso termina pulsando un botón de finalización. El procedimiento de devolución de libros funciona de un modo análogo.

Para ser admitido en la biblioteca como socio, un usuario tiene que ser dado de alta por el personal de la biblioteca. A partir de los datos del usuario el sistema genera una tarjeta de socio.

En la puerta de salida de la biblioteca existe un dispositivo de seguridad; éste se activa bloqueando las puertas en el momento en el que un socio intenta salir del edificio con un libro que tenga el código antirrobo activado. Pulsando un botón de emergencia las puertas se desbloquean.

El personal de la biblioteca se encarga de las labores de mantenimiento de libros y de socios, y de las consultas relativas al estado de los socios y los libros. Los sócios pueden realizar búsquedas sobre el catálogo disponible (por autor, título, palabras clave, etc.). Existe un tratamiento uniforme con respecto a la política de préstamos: cada libro que no se devuelva en el plazo estipulado originará una sanción al socio correspondiente.

Ej. 13. Sobre el problema del ejercicio anterior, describa al menos tres escenarios de uso (sin descuidar casos excepcionales) mediante casos de uso.

Ej. 14. Considere el siguiente fragmento de código:

```
int n,
    lcv,
    flag; /* flag initially is 1 and becomes 0 if we determine that n
     is not a prime */

 for (lcv=2, flag=1; lcv <= (n / 2); lcv++) {
    if ((n % lcv) == 0) {
      if (flag)
printf("The non-trivial factors of %d are: \n", n);
      flag = 0;
      printf("\t%d\n", lcv);
    }
 }
 if (flag)
    printf("%d is prime\n", n);
```

Construya conjuntos de casos de test para el mismo que cumplan con los criterios de cobertura de sentencias y de ramas, y sabiendo que la variable de entrada es n, de tipo entero.