DESARROLLO DE INTERFACES

UF2 PAC1 - DOCUMENTACIÓN DE APLICACIONES. REALIZACIÓN DE PRUEBAS. DISTRIBUCIÓN DE APLICACIONES.

Aníbal Santos Gómez

Parte teórica

1. Responde a este test en la matriz de respuestas que se encuentra al final de las preguntas:

1.1. Las pruebas de caja blanca cubren campos tales como:

a. Cobertura de sentencias, cobertura de decisiones
b. Cobertura de condiciones
c. Cobertura de caminos.
d. Todas las anteriores.

1.2. Las pruebas de caja negra cubren campos tales como:

a. Cobertura de sentencias, cobertura de decisiones
b. Cobertura de condiciones
c. Cobertura de caminos.
d. Ninguna de las anteriores.

1.3. Podemos definir prueba de sistema como:

a. Aquellas que buscan asegurar que el código de acuerdo con las especificaciones definidas y que el módulo lógico es válido. La operativa de este tipo de pruebas se centra en ejecutar cada módulo (o unidad mínima a ser probada, como por ejemplo una clase) lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores.

b. Aquellas en las que se prueban los diferentes módulos o componentes
de forma conjunta a fin de encontrar errores que pudieran ser producidos por los efectos de uno sobre otro.

c. Aquellas que tienen como objetivo testear de forma completa todo el sistema, de forma que compruebe la correcta integración de todos los elementos y que opera de forma correcta de acuerdo con las especificaciones definidas.

d. Aquellas que consisten en la repetición selectiva de pruebas para detectar fallos introducidos durante la modificación de un sistema o

componente del sistema.

1.4. Podemos definir prueba alfa como:

a. Aquellas que tienen por objetivo probar que el sistema desarrollado cumple con las funcionalidades específicas para las que ha sido creado. En este tipo de pruebas participan analistas y usuarios finales.

b. Aquellas que están encaminadas a ver cómo responde el sistema en ciertas condiciones de trabajo, elevado número de usuarios, conexiones simultáneas a la base de datos, etc. Es decir, simula el comportamiento de la aplicación.

c. Aquellas que se llevan a cabo en el entorno del desarrollador, mediante un grupo representativo de usuarios finales, esto es, en un entorno controlado bajo la supervisión del desarrollador.

d. Aquellas que se realizan cuando el producto está listo para implantarse en el entorno cliente. Estas pruebas deben ser realizadas por usuarios finales.

1.5. Podemos definir prueba beta como:

a. Aquellas que tienen por objetivo probar que el sistema desarrollado cumple con las funcionalidades específicas para las que ha sido creado. En este tipo de pruebas participan analistas y usuarios finales.

b. Aquellas que realiza el usuario final en un entorno no controlado y sin la supervisión del desarrollador.

c. Aquellas que se llevan a cabo en el entorno del desarrollador, mediante un grupo representativo de usuarios finales, esto es, en un entorno controlado bajo la supervisión del desarrollador.

d. Aquellas que se realizan cuando el producto está listo para implantarse en el entorno cliente. Estas pruebas deben ser realizadas por usuarios finales.

1.6. Los elementos textuales que aparecen en la interfaz al posicionarse el usuario con el puntero del ratón durante un par de segundos sobre el elemento se conoce como:

a. HelpTip.
b. **ToolTip.**
c. ElemTip.
d. Ninguna de las anteriores.

1.7. Durante el desarrollo de un proyecto de software se generan los siguientes tipos de documentación:

a. Manual de instalación, Manual de usuario, Manual de configuración.
b. Guía de usuario, guía rápida
c. Manual de administración
d. Todas las anteriores.

MATRIZ DE RESPUESTAS:

1. D
2. D
3. C
4. C
5. B
6. B
7. D

2. Realiza una descripción detallada de los tipos de pruebas existentes. Puedes ayudarte de tablas, esquemas, dibujos, diagramas.

Los tipos de prueba a describir son los siguientes:

- **Prueba unitaria.**

La prueba unitaria busca asegurar que el código está funcionando según las especificaciones definidas. Se ejecuta cada módulo independientemente.

La prueba se lleva a cabo mediante el particionamiento de los módulos en pruebas en unidades lógicas, para cada unidad hay que definir los casos de prueba, los casos de prueba tienen que diseñarse de tal forma que se recorran los caminos de ejecución.

Los aspectos destacables son, las rutinas de excepción, las rutinas de error, el manejo de parámetros, validaciones, valores válidos y límites, rangos y mensajes posibles.

Como objetivo debemos comparar el resultado esperado con el resultado obtenido.

- **Prueba de integración.**

Esta prueba consiste en probar los componentes de forma conjunta, o bien de forma **ascendente**, combinando los módulos de bajo nivel en grupos que realicen una subfunción, diseñando un gestor para coordinar la entrada y salida de los casos de prueba, comprobando el funcionamiento del grupo y eliminando posteriormente los gestores y combinándose hacia arriba por la estructura del programa.

O bien de forma **descendente**, usando el módulo de control principal como controlador de prueba, creando resguardos para todos los módulos directamente subordinados al módulo de control, se van sustituyendo uno a uno los resguardos subordinados por los módulos reales y se llevan a cabo pruebas cada vez que se integra un nuevo módulo.

- **Prueba de sistema.**

La prueba de sistema testea completamente el sistema, se comprueba la integración del hardware.

Su objetivo principal es evaluar los siguientes aspectos; que se cumplan los requisitos funcionales definidos, el funcionamiento y el rendimiento de las interfaces de hardware, software y usuario, la adecuación de la documentación de usuario, el rendimiento y las respuestas en condiciones límite y de sobrecarga, y por último asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación.

Para generar pruebas, utilizamos pruebas de caja negra, se realizan en el entorno de desarrollo y se denominan pruebas alfa, después se llevan a cabo en el entorno del cliente, denominándolas pruebas beta.

Se llevan a cabo pruebas de configuración donde se busca testear el sistema en diferentes entornos de configuración hardware-software. Aquí aseguramos que la aplicación alcanzará sus objetivos de negocio.

Las pruebas incluidas en las pruebas de configuración de sistema son de funcionalidad, usabilidad, performance, documentación, seguridad, volumen, esfuerzo y recuperación.

- **Prueba de regresión.**

Cuando agregamos un nuevo módulo o componente incluimos nuevas rutas de flujo de datos y lógica de control de software, esto puede afectar en el funcionamiento normal del módulo provocando errores en lo que antes funcionaba.

La prueba de regresión consiste en la repetición selectiva de las pruebas para descartar fallos introducidos durante la modificación. En dichas pruebas hay que probar íntegramente los módulos que han cambiado y decidir si las pruebas a efectuar para los módulos que no han cambiado y han sido afectados por los cambios efectuados.

- **Prueba funcional.**

El cometido de dicha prueba es probar el sistema desarrollo. Empleamos equipos formados por analistas de pruebas y usuarios finales. Se lleva a cabo en la fase final del proyecto y permite decidir si se pasa al entorno de producción en caso de ser satisfactoria.

- **Prueba de capacidad y rendimiento.**

Estas dos pruebas están orientadas a la observación de la respuesta del sistema en condiciones de trabajo. Estas pruebas pueden ser:

- De humo, para cargas ligeras de trabajo.
- De tensión, para cargas pesadas en tiempo continuado.
- De rendimiento.
- De diseño de capacidad que determina el comportamiento de la aplicación bajo distintas capacidades.

Además tenemos que tener en cuenta el proceso de carga de datos mediante los diferentes modelos:

- Modelo constante, que establece una carga idéntica de un usuario durante un tiempo constante.

- Modelo de paso, que supervisa la aplicación cuando se aumentan los valores de carga. Con ello obtenemos los umbrales de tiempo de respuesta de la aplicación en parámetros aceptables.

- Modelo de objetivos, que nos permitirá determinar la detención sobre el aumento de carga.

- **Prueba de uso de recursos.**

Aquí determinaremos los elementos críticos para el sistema, que por lo general son la CPU, el espacio disponible en el disco de almacenamiento y la memoria RAM.

- **Prueba de seguridad.**

Mediante dicha prueba trataremos de asegurarnos que los mecanismos que hemos establecido y definido sean correctos para evitar los accesos no permitidos.

- **Prueba de usuario.**

A través de la prueba de usuario comprobaremos la usabilidad de la aplicación. Para ello escogeremos a un grupo de usuarios con diferentes perfiles y habilidades. Mediremos el tiempo y el número de clicls que emplea dicho usuario en realizar una tarea determinada.

En estos ensayos se podrán realizar encuestas y entrevistas, grupos de enfoque, pruebas de laboratorio y evaluaciones heurísticas.

- **Prueba de aceptación.**

En este momento el producto está listo para implantarse en el entorno cliente. El usuario realizará las pruebas ayudado por los componentes del equipo de pruebas.

Estas pruebas se caracterizan por la participación activa del usuario, están enfocada a probar los requisitos de usuario, corresponden a la fase final del proceso de desarrollo.

Aquí distinguimos entre **pruebas alfa**, que se llevan a cabo en el entorno del desarrollador, mediante un grupo representativo de usuarios finales, y **pruebas beta**, donde el usuario carece de la supervisión mencionada y se encuentra en un entorno no controlado, similar al que tenía en producción.

3. Realiza una descripción detallada de los tipos de documentos que suelen acompañar a la creación de un proyecto. Puedes ayudarte de tablas, esquemas, dibujos, diagramas.

Los documentos que suelen acompañar a la creación de un proyecto son:

- **Manual de instalación:**

En este documento encontraremos la información necesaria para instalar la aplicación en diferentes entornos. Se describen aspectos del software como los sistemas operativos y los requisitos mínimos de hardware y software.

Existen diferentes posibilidades de instalación explicadas como, el cambio de directorios por defecto, generación de archivos log, ubicación de archivos de configuración, tipo de instalación o apariencia.

Además suele haber dos tipo de versiones de este manual, el detallado, donde se explican ampliamente cada uno de los aspectos anteriores, y la reducida, donde se explica una instalación rápida sin demasiados detalles.

- **Documentación de configuración:**

En la documentación de configuración se explican los parámetros de configuración de la aplicación. Se especifica como realizar cambios y sus consecuencias a nivel del programa. Existen dos niveles de configuración, normal o avanzada.

- **Manual de usuario:**

Este documento se realizará teniendo en cuenta el nivel de los usuarios que usarán la aplicación, por lo que seguramente deberemos realizar varias versiones, desde el principiante al experto.

Dependiendo del nivel habrá un grado de profundización mayor o menos en las explicaciones. Estos detalles aumentarán a medida que el nivel del manual aumenta.

- **Guía de usuario:**

Tiene como objetivo incidir de forma rápida en las funcionalidades del producto de forma que el cliente pueda operar de forma inmediata.

- **Guía rápida:**

La guía rápida detalla sintetizadamente la forma de realizar una tarea.

- **Manual de administración:**

Este manual nos permite conocer los detalles de la instalación, la administración del sistema y la configuración de los componentes de la aplicación.

Parte práctica

4. Crea una prueba unitaria (en Visual Studio) que evalúe una subrutina que calcule el factorial de un número.

El código del programa es el siguiente:

- **Clase Program (donde está el el método Main):**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UF2_PAC1
{
```

```
        class Program
        {
        static void Main(string[] args)
        {
        int num;
        Console.WriteLine("Este programa calcula el factorial del número.\n");
        Console.WriteLine("Por favor, introduzca un número:\n\n ");
        num = int.Parse(Console.ReadLine());
        Class1 fac = new Class1();

        int res = fac.FactorNum(num);

        Console.WriteLine("\n\nEl factorial de " + num + " es: " + res);
        Console.WriteLine("\n\nPulsa cualquier tecla para salir del programa.");
        Console.ReadKey();
        }
        }
}
```

- **Clase de la función que calcula el factorial introducido por el usuario:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UF2_PAC1
{
        public class Class1
        {
        public int FactorNum (int nu)
        {
        int re = nu;
        for (int i = 1; i < nu; i++)
        {
                re = re * i;
        }
        return re;
```

```
        }
        }
}
```

- **Test unitario de la función de la clase Class1:**

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using UF2_PAC1;

namespace UnitTestProject1
{
        [TestClass]
        public class UnitTest1
        {
        Class1 objeto = new Class1();
        [TestMethod]
        public void TestMethod1()
        {
        //Resultado -- 120

        int expected = 120;

        //Resultado del codigo

        int result = objeto.FactorNum(5);

        //Assert;

        Assert.AreEqual(expected, result);
        }
        }
}
```

Se adjunta el programa completo comprimido.