Módulo Profesional 05:
Entornos de desarrollo

UF2 Actividad A

CICLO FORMATIVO DE GRADO SUPERIOR EN
DESARROLLO DE APLICACIONES MULTIPLATAFORMA
MODALIDAD ONLINE

ALUMNA:

MARÍA LAURA RONDINA IOBBI

Descripción de la actividad:

En la siguiente actividad teórica se valorarán los conceptos más importantes de la actividad. Por lo que deberás de dar respuesta a los siguientes ejercicios teóricos en el mismo documento y entregar por el campus. Poner nombre al documento y entregar en formato Word/pdf.

Dar respuesta a los siguientes ejercicios teórico en el mismo documento para entregar por el campus.

1. ¿Cuáles son los dos aspectos fundamentales que debe de tener una batería de pruebas? [0,5 puntos]

La realización de pruebas es un aspecto vital en la actividad del desarrollo de software. Debemos partir de la base de que resultaría imposible considerar todas las circunstancias de datos posibles que un programa afrontará cuando sea trasladado a un entorno real de explotación; no obstante, al diseñar una batería de pruebas se deben contemplar dos aspectos fundamentales para minimizar la probabilidad de errores al trasladarlo al entorno real:

a) **Casuísticas Habituales**: las pruebas que simulen las casuísticas más habituales que el programa procesará.

b) **Casuísticas Críticas**: aquellas situaciones críticas en las que un fallo del producto podría generar un grave problema.

La valoración de los aspectos “cotidianos” (que permiten cubrir un alto porcentaje de situaciones) y los “críticos” (para evitar circunstancias indeseables y que producirían un significativo coste económico a la empresa) son los dos puntos guía para realizar las pruebas y poder progresar en un entorno real de explotación con el máximo de garantías posibles.

2. ¿Qué aspectos se deben de tener en cuenta en la planificación de un periodo de pruebas? [0,5 puntos]

Planificar correctamente un período de pruebas comporta tener en cuenta los siguientes aspectos:

● **Calendario**. Cuándo y durante cuánto tiempo podemos hacer pruebas.

● **Normas.** Objetivos de la compañía, ámbito de las pruebas, metodología a utilizar, etcétera.

● **Recursos.** Hardware y software necesarios para los test, herramientas que van a utilizarse y perfiles de los técnicos y usuarios a los que se les requerirá su colaboración.

● **Productos.** Incluye todos los elementos que conforman la futura instalación en un entorno real.

● **Contingencia.** A pesar de que no siempre es posible tener en cuenta todas las circunstancias por las que puede desarrollarse una prueba de producto, es importante considerar que las cosas pueden no salir como se espera y es por eso que en caso de fallar la prueba principal del producto se deben tener alternativas de solución .

3. ¿Cuáles son los 3 principales tipos de pruebas que hay? Defínelos. [0,5 puntos]

Una vez finalizado el desarrollo del producto, es necesario comprobar su funcionamiento. Para ello, se debe realizar una serie de test que lo pongan a prueba y verifique que los resultados no son aleatorios o motivados por otras causas que no sean las propias de las funcionalidades a instalar. Los tres principales tipos de pruebas que hay son las siguientes:

1) Prueba funcional: Este tipo de pruebas son externas al código del programa, es decir, sólo miden resultados. Comprueban si el producto hace realmente lo que se espera que haga; trata de comparar las exigencias del cliente y la solución ofrecida por el analista a través del diseño técnico con los resultados del software. Para ello, se utilizan unos datos de test preparados para tal efecto y un caso de prueba para que, antes de que el software nos proporcione el resultado, éste se conozca a priori para poderlo valorar.

2) Prueba estructural: Es la que se adentra en el propio programa y, a partir del estudio del código, de su estructura interna y su confrontación con el diseño técnico, llega a las conclusiones necesarias para tomar las acciones correctoras que fueran necesarias. Este tipo de prueba utiliza casos de prueba relacionados con la lógica del código fuente, estructuras de datos, tablas, tiempos de ejecución y control correcto de bucles de sentencias. En el caso de que sea un software configurable por el usuario, se debe considerar el control que se haga de selecciones imposibles o que puedan llevar a errores de computación. Dicha prueba estructural no debería, en teoría, realizarla el propio programador, aunque con frecuencia así se haga, rellenando un documento de test técnico.

3) Prueba de regresión: Esta prueba se aplica en el mantenimiento de un software ya existente al que se le quiere realizar una modificación, ya sea porque contenía un error o porque se incorpora una mejora al producto. En caso de ser un producto nuevo, la prueba de regresión se utiliza después de haber realizado el test principal, e implica la realización de cambios. En ese caso, este tipo de prueba intenta probar que nuestra modificación no ha generado nuevos errores o influye negativamente en la ejecución normal del programa. La forma más sencilla de esta prueba consistiría en crear un “caso de prueba” y probar el programa dos veces: una sin la modificación, y la otra con la modificación, para comprobar el impacto que ha tenido y corregirlo si es necesario.

4. ¿Qué son las herramientas de depuración? [0,5 puntos]

Las **herramientas de depuración** son unos programas cuyo objetivo principal es el de probar el funcionamiento de otros programas informáticos. Para ello se utiliza una serie de acciones, entre las que caben destacar:

● **Puntos de ruptura (Breakpoint).** El programador puede poner en el código fuente unas señales que la herramienta de depuración interpretará como un punto de parada de ejecución. Cuando el control del programa llegue a ese punto, el programa se detendrá temporalmente, pero mantendrá los valores de variables, funciones y objetos en la memoria. Esto permitirá al programador conocer el valor o estado de cada elemento que deberá comprobar y modificar si fuese necesario. Una vez analizados los datos, es posible continuar la ejecución a partir de dicho punto.

● **Tipos de ejecución.** Según la naturaleza de lo que pretendamos comprobar, podremos utilizar uno de los siguientes métodos: - **Ejecución línea a línea**. Se emplea cuando el problema lo tenemos muy delimitado o deseamos una ejecución detallada. Por cada línea de código, el programa se detiene. - **Ejecución por grupos**. Mientras no termina una declaración completa de sentencias, el programa no se detiene.

● **Examinadores de variables.** Las ventanas de salida en modo “depuración” presentan los valores que tienen las variables en un momento concreto. A partir de un punto de interrupción también se puede interrogar al entorno de desarrollo acerca de un valor determinado. Existen, además, otras

formas dinámicas de examinar variables, que provocan puntos de interrupción cuando una variable tiene un valor marcado por el programador.

5. ¿Qué son las normas de calidad? [0,5 puntos]

Las **normas de calidad** son documentos publicados por organismos competentes en los que se especifican los requisitos, las reglas y las formas de trabajo que una empresa debe seguir para poder ser considerada "certificada" en dicha norma. Las empresas de software también pueden necesitar certificarse en calidad, por varias razones: porque el auditor que compruebe si la empresa sigue la normativa la corregirá mediante "no conformidades" que deberá eliminar con el tiempo; porque en un sistema económico globalizado como el actual, se exige poseer unas conductas de trabajo acordes con los requisitos del cliente; y porque los futuros accionistas desearán conocer en qué tipo de compañía están invirtiendo su dinero. La normativa de calidad más común comprende:

● **ISO/IEC 15504 SPICE - Software Process Improvement Capability Determination**. Este grupo de normas facilita la mejora continua de los procesos de las organizaciones, especialmente aquellos relacionados con el desarrollo y mantenimiento del software. Se basa en la evaluación a través de niveles de madurez de la empresa, entendiendo por éstos el conjunto de procesos predefinidos que ayudan a una organización a mejorar en el desarrollo del software. Estos niveles son: - **Nivel 0**: Inmadura. No hay implementación de procesos. -**Nivel 1**: Básica. Se alcanzan los objetivos de los procesos. - **Nivel 2**: Gestionada. Gestión de procesos y productos. - **Nivel 3**: Establecida: Procesos adaptados/estándares. - **Nivel 4**: Predecible: Gestión Cuantitativa. - **Nivel 5**: Optimizado. Mejora continua de procesos.

● **ISO 9001**. Es la solución más eficaz para las organizaciones que pretendan el desarrollo e implantación de Sistemas de gestión de calidad, además de abarcar otros ámbitos de la empresa, como el servicio al cliente, sistemas de mantenimiento y flujos administrativos, entre otros.

● **ISO IEC 90003 2004.** Esta normativa sirve de guía para una constante mejora. Se aborda desde diferentes ámbitos: - Requerimientos del sistema empresarial. - Requerimientos de gestión. - Requerimientos de recursos. - Requerimientos de control y realización. - Requerimientos de contingencia y reparación.

● **Normativa IEEE 829-2008**. Proporciona un conjunto estandarizado para la documentación de test de sistemas y software. Las tareas del proceso de prueba se especifican para diferentes niveles de integridad. Su alcance abarca sistemas basados en software, hardware y sus interfaces. Esta norma es de aplicación a los sistemas basados en software que se desarrollan, mantienen o reutilizan elementos heredados, comerciales y no evolutivos.

6. ¿Qué es una prueba unitaria? [0,5 puntos]

Una **prueba unitaria** es aquella que se realiza en un sólo componente del programa de forma aislada. Dicho componente puede ser un módulo o una clase, si bien el concepto de unidad será manipulado por el programador para aplicarlo allí donde considere oportuno. El aislamiento del componente del resto de la aplicación es, en ocasiones, complicado, ya que éste puede depender de las interacciones que tenga el componente analizado con el resto. Algunas herramientas que facilitan dicha tarea son:

● **JUnit.** Conjunto de clases que asiste a los desarrolladores de Java para comprobar si un programa funciona correctamente. En caso de que el valor devuelto por un proceso no se corresponda con el esperado, es posible integrar JUnit en entornos de desarrollo, como NetBeans o Eclipse, a través de los complementos o plugins.

● **PHPUnit.** Orientado a la creación de juegos de pruebas unitarias en lenguaje PHP.

● **NUNit.** Un gestor de pruebas unitarias para los lenguajes .NET. La propia herramienta está escrita en C#.

7. Investiga en que consiste la normativa IEEE 829-2008, también conocida como la norma 829 para la documentación de test de sistemas y software. Marcar si los siguientes contenidos pertenecen a (V) o no (F) al plan de pruebas, según la normativa IEEE 829-2008. [5,0 puntos]

Pregunta	V	F
Identificador del plan de pruebas	**V**	
Descripción del plan de pruebas	**V**	
Descripción del proyecto		**F**
Elementos del software que no se han de probar	**V**	
Definición de la configuración del plan de pruebas	**V**	
Requisitos del proyecto		**F**
Documentos a liberar	**V**	
Responsables y responsabilidades	**V**	
Partes interesadas (o stakeholders)		**F**
Calendario del plan de pruebas	**V**	
Calendario de la fase de desarrollo		**F**

8. Relaciona los tipos de pruebas con los objetivos o características que las definen. [2,0 puntos]

(**d**) Son las encargadas de detectar los errores en la implementación de los requerimientos de usuario	a. Tipos de pruebas de aceptación
(**c**) El método utilizado es de caja blanca, el de diseño descendiente (o top-down) y el de bottom-up	b. Tipos de pruebas de sistemas
(**b**) Su finalidad es detectar errores en la obtención de los requerimientos.	c. Tipos de pruebas de integración
(**a**) Su objetivo es la validación de la aplicación por parte de los usuarios.	d. Tipos de pruebas funcionales