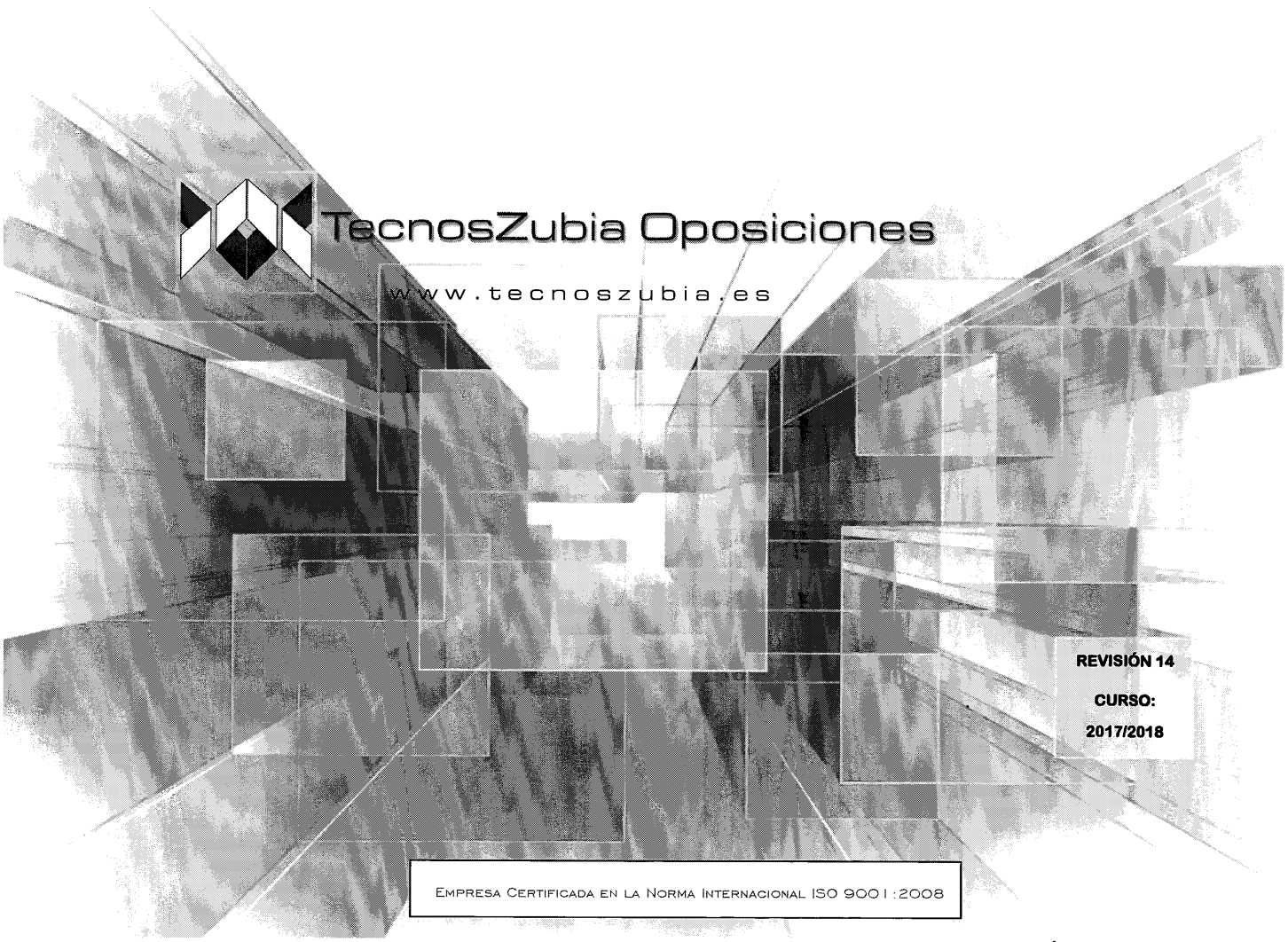
INFORMÁTICA

TEMA 47:

Instalación y explotación de aplicaciones informáticas. Compartición de datos.



TecnosZubia Oposiciones

www.tecnoszubia.es

REVISIÓN 14

CURSO:

2017/2018

EMPRESA CERTIFICADA EN LA NORMA INTERNACIONAL ISO 9001:2008

TEMA 47. INSTALACIÓN Y EXPLOTACIÓN DE APLICACIONES INFORMÁTICAS. COMPARTICIÓN DE DATOS.

1. INTRODUCCIÓN

Este tema se puede abordar de distinta forma dependiendo de si consideramos una aplicación informática comercial de propósito general, o una de propósito específico es decir, diseñada para un sistema de información en concreto.

En este capítulo nos vamos a centrar en el segundo caso.

En aplicaciones informáticas encargadas a una empresa de software para la automatización de un sistema de información, el proceso de instalación de la aplicación es realizado por dicha empresa, ya que está contemplada esta tarea como una etapa más en el ciclo del desarrollo del software.

En concreto esta tarea corresponde a la última etapa, y tiene como finalidad la puesta en marcha del sistema desarrollado y conseguir la aceptación de éste por parte de los usuarios del mismo, probando el sistema completo en un entorno de trabajo real.

Dentro de las fases del ciclo de vida de un proyecto en Ingeniería del Software, esta última etapa se denomina **Implantación del Sistema**.

Para la realización de esta fase, se toma como punto de partida la documentación de las fases anteriores.

Existen diversas **estrategias de implantación** que pueden seguirse, dependiendo su elección de múltiples factores, como por ejemplo:

- El tipo de sistemas que puedan solaparse con el sistema a implantar.
- El grado de descomposición que permita una implantación en fases, durante un período de tiempo.

En ocasiones, será necesario realizar la **transición del sistema antiguo al nuevo** de forma instantánea. Sin embargo, siempre que los horarios y los recursos lo permitan, el funcionamiento en paralelo de ambos sistemas puede utilizarse para minimizar el riesgo. En este último caso, uno de los dos sistemas se considerará el "Maestro":

- Si se elige el sistema existente como el sistema maestro, la operación en paralelo del nuevo sistema podría ser considerado como una sustitución a las pruebas de aceptación.
- Si se elige como sistema maestro al nuevo sistema, el sistema existente proporciona una seguridad adicional ante un posible error del nuevo.

2. PRUEBAS

Las pruebas a realizar se van a dividir en dos grandes bloques: Pruebas de Sistemas y Pruebas de Aceptación, ambas son similares en cuanto a lo que van a probar, sin embargo, el entorno técnico donde tendrán lugar es distinto:

- **Las Pruebas de Sistema** cubren un rango muy amplio, que va desde la comprobación de cualquier detalle de diseño interno hasta aspectos tales como las comunicaciones.

- **Las Pruebas de Aceptación** se realizan por y para los usuarios y tienen como objeto conseguir una demostración formal de que el sistema cumple todos los requisitos de usuario.

2.1. Pruebas del Sistema

En la especificación de las pruebas se van a tener que contemplar los siguientes aspectos:

1. **Comunicaciones**, para determinar el correcto funcionamiento de las interfaces entre componentes (incluyendo enlaces entre dispositivos).

2. **Rendimiento** en la ejecución. Se medirá el tiempo de respuesta del sistema bajo diferentes condiciones.

3. **Volumen**. Se comprobará el sistema actuando sobre grandes volúmenes de datos. Habrá que tener en cuenta el comportamiento en las condiciones críticas.

4. **Recuperación de datos**. Se probará que el sistema puede recuperar datos, cuando haya fallos del equipo.

5. **Operaciones**. Se comprobará el funcionamiento adecuado de los procedimientos de arranque y finalización del sistema.

6. **Seguridad.** Se verificará que los mecanismos de seguridad definidos son suficientes.

7. **Integración** con productos de aplicaciones existentes o que hayan sido adquiridos en la Unidad.

Se prepararán todos los procedimientos y facilidades necesarias para la realización de pruebas del sistema.

Una vez realizadas las pruebas, según las especificaciones diseñadas previamente, se documentarán los resultados de las mismas en un "Informe de Pruebas del Sistema".

Dependiendo del resultado de estas pruebas, se emprenderán las acciones correctivas necesarias, para solucionar los problemas e incidencias ocurridos durante la ejecución de las mismas.

2.1.1. Preparación de las pruebas del sistema

El principal objetivo de esta tarea, es revisar y completar los planes de prueba del sistema, definidos en fases anteriores. Además se definirán pruebas adicionales, basadas en los Procedimientos de Usuario.

Las especificaciones de prueba deberán incluir:

- **Un Plan de Pruebas del Sistema**, donde se recogen los objetivos de las pruebas del sistema, entre los que se encuentran:

    1. Comprobar que el sistema se ajusta a las especificaciones funcionales del mismo.
    2. Comprobar la adecuación de los procedimientos de usuario al funcionamiento del sistema.
    3. Comprobar la respuesta del sistema.
    4. Verificar que el sistema responde con precisión, en situaciones de altos volúmenes de información, y dentro de unos límites ajustados de tiempo de respuesta
    5. Comprobar el funcionamiento correcto del sistema en lo que respecta a:
        Captura de información.
        Validación y edición.
        Tratamiento de errores.
        Actualización de la información.
        Obtención de informes.
        Conversión de la información.
        Interfaces entre programas.
        Interfaces externas.
        Procedimientos de copias de respaldo y de recuperación.

- **Las especificaciones de los casos de prueba**, los cuales han de contemplar datos actuales y datos especialmente generados para satisfacer alguna de las pruebas.

- **Procedimientos** detallados para la ejecución de cada prueba.

2.1.2. Creación del entorno de pruebas del sistema

Los pasos a seguir para la Creación del Entorno de Pruebas del Sistema son:

1. Refinar los procedimientos de control de versión del equipo lógico.
2. Crear las bases de datos o ficheros de prueba definidos en la fase anterior.
3. Instalar, en su caso, las librerías que precisen las pruebas del sistema.
4. Crear los procedimientos de gestión de pruebas.
5. Probar todas las facilidades necesarias para la ejecución de las pruebas (terminales, comunicaciones,...).
6. Probar las bases de datos o ficheros que se utilizarán, y los procedimientos de respaldo, recuperación y acceso.
7. Verificar los procedimientos de prueba:
   - Iniciar las sesiones de prueba.
   - Recibir o producir copias de los resultados.
   - Mantener las librerías de programas de pruebas.

2.1.3. Realización de las pruebas del sistema

La realización de las pruebas del sistema se llevará a cabo siguiendo los siguientes pasos:

1. Instalar los "componentes" necesarios para las pruebas.
2. Preparar la base de datos o ficheros de prueba, iniciar los valores necesarios para la ejecución de las pruebas y crear las tablas y ficheros auxiliares que sean precisos.

3. Cargar los datos necesarios para las pruebas, y recopilar los datos manuales, proporcionados por los usuarios, que deben ser probados.

4. Ejecutar cada una de las pruebas, siguiendo lo definido en la fase de Preparación de las Pruebas del Sistema.

5. Documentar todas las incidencias ocurridas durante la ejecución de las pruebas

6. Evaluar los resultados de las pruebas y resolver los problemas derivados:

    - Determinar las acciones que deben llevarse a cabo para resolver los problemas detectados. Las acciones deberán enfocarse hacía el módulo donde se introdujo el error y se realizarán las actividades correspondientes.

    - Tras realizar las oportunas acciones correctivas, se actualizará toda la documentación.

    - Determinar el conjunto de pruebas del sistema que deben volver a realizarse, ejecutarlas y comprobar el impacto de los cambios efectuados.

7. Verificar y validar el conjunto de pruebas del sistema.

8. Completar el Informe de las Pruebas del Sistema:

    • Determinar si las pruebas pueden considerarse completas.

    • Notificar los problemas no resueltos.

    • Elaborar un informe final

2.2 Pruebas de Aceptación

La realización de las pruebas de aceptación buscan que el usuario apruebe el nuevo sistema, además de demostrar a los usuarios que dicho sistema satisface todos sus requisitos, los cuales fueron determinados y especificados en la fase de análisis de requisitos del sistema y en la especificación funcional del sistema.

La **Especificación de las Pruebas de Aceptación** deberá constar de:

- Especificación detalladas de las **pruebas de Aceptación**
- **Datos** que van a usarse en los casos de prueba. Los casos de prueba que se especifican en esta actividad han de permitir valorar los siguientes aspectos del sistema:
  1. Verificación de las salidas del sistema.
  2. Calidad y exactitud de los datos convertidos para su uso por el nuevo sistema.
  3. Medida del cumplimiento de los requisitos y funciones especificas.
  4. Adecuación de los procedimientos de usuario e instrucciones de operación del sistema.
  5. Comportamiento adecuado del sistema en situaciones de sobrecarga.
  6. Eficiencia en el tratamiento realizado por el sistema.

El nuevo sistema puede haberse desarrollado en un entorno distinto de aquél en el que finalmente se implantará.

Las pruebas de aceptación deben llevarse a cabo en el entorno real de producción, por lo que será también objetivo de esta actividad comprobar que este entorno está instalado y verificado.

El entorno de producción incluye los equipos físicos, el equipo lógico del sistema y de aplicación y los procedimientos de operación.

2.2.1. Preparación de las pruebas de aceptación

En esta fase se pretende revisar y corregir las especificaciones de las Pruebas de Aceptación y preparar los datos necesarios para desarrollar los casos de pruebas de aceptación.

Habrá que tener en cuenta que la principal diferencia entre las Pruebas de Sistema y las de Aceptación es que las últimas son realizadas por los usuarios, por lo que estas pruebas comprueban si se verifican los requisitos de dichos usuarios.

Seguida de esta actividad, está la Preparación del Entorno de Producción.

Para ello se comprobarán una serie de recursos físicos y humanos, de forma que entre las tareas a realizar se encuentran aquellas de comprobación del material físico y lógico necesario, así como comprobar que todos los usuarios participantes en las pruebas han recibido la formación precisa para una correcta ejecución de las pruebas.

2.2.2. Realización de las pruebas de aceptación

Los pasos a seguir para el desarrollo de esta tarea son los siguientes:

1. Instalar el equipo lógico del sistema a probar, así como los datos necesarios en el entorno de producción.
2. Realizar todos los procedimientos definidos para la ejecución de pruebas de aceptación.
3. Completar un informe de la ejecución de las pruebas.
4. Evaluar los resultados de las pruebas y resolver los problemas derivados.
5. Completar un informe para cada una de las pruebas realizadas teniendo en cuenta: la evaluación de cada componente del sistema, las desviaciones detectadas sobre los requisitos y problemas sin resolver.
6. Conseguir la aceptación del sistema.

3 INSTALACIÓN DE APLICACIONES INFORMÁTICAS

3.1 Introducción

Como dijimos en la introducción del presente tema, dentro de las fases del ciclo de vida de un proyecto en Ingeniería del Software, esta última etapa se denomina **Implantación del Sistema**.

Para la realización de esta fase, se toma como punto de partida la documentación de las fases anteriores.

Los **objetivos** fundamentales de la fase de implantación son:

- Verificar que funcionalmente se cumple con todos los requisitos que se determinan en la fase de análisis de requerimientos.
- Comprobar que el sistema es capaz de manipular los volúmenes de información requeridos, de trabajar dentro de los tiempos de respuesta deseados y que funcionan correctamente los procedimientos de copias de respaldo, seguridad e interfaces con otros sistemas. El comportamiento debe examinarse bajo las condiciones más extremas.

3.2 Plan de Implantación

En general, la implantación de un sistema puede considerarse como un conjunto de actividades necesarias para añadir, modificar o crear los datos necesarios para el funcionamiento correcto del nuevo sistema, e instalar tanto los componentes del nuevo sistema, como el conjunto de procedimientos manuales y automáticos requeridos para él.

Los principales aspectos a considerar dentro del Plan de Implantación son aquellos relacionados con la planificación de la conversión de datos (si la hubiera), de los procedimientos de instalación, de la instalación de productos estándar adquiridos externamente, plan de contingencias, planes de horario y trabajo y acciones de revisión, que se llevarán a cabo una vez realizada la implantación.

El Plan de Implantación cuenta con una serie de tareas como son su revisión y preparación. A través de estas tareas, partiendo de la información obtenida inicialmente, y mediante la información que se va almacenando durante el desarrollo del proyecto se irá modificando, si fuera necesario, dicho Plan de Implantación.

3.3 Proceso de Instalación

Lo primero a tener en cuenta al instalar cualquier aplicación informática son los requisitos de dicha aplicación.

Una vez que comprobamos que nuestro equipo cumple con todos los requisitos necesarios y hemos adquirido el paquete, procederemos a su instalación. Una vez instalada la aplicación, realizaremos en ella los cambios en la configuración que se deseen, para que su funcionamiento satisfaga nuestra necesidades concretas.

3.3.1 Requisitos previos a la instalación:

Este tipo de requisitos suele recomendar una configuración hardware mínima, como tipo de plataforma para la que está diseñada (por ejemplo PC o MAC), velocidad mínima del sistema recomendada, cantidad de memoria RAM, cantidad de espacio en disco duro, etc. Y una serie de recomendaciones opcionales que pueden ser útiles para sacar el mayor partido a la aplicación.

3.3.2 Puesta en marcha del programa de instalación

El siguiente paso en la instalación es la puesta en marcha del programa de instalación, que básicamente descomprimirá los ficheros incluidos en el paquete y los instalará en el disco duro del ordenador. Este programa ofrecerá al usuario la posibilidad de realizar una instalación típica o una instalación personalizada, opción que normalmente se recomienda a usuarios expertos.

- La **instalación típica** instalará los componentes más usuales de la aplicación en el directorio que traiga por defecto creando, al finalizar, accesos directos a la aplicación para facilitar al usuario la ejecución.

- La **instalación personalizada** dará opción a instalar los componentes de la aplicación que al usuario le parezca oportunos. En este sentido el usuario podrá elegir:

    - Instalar un componente en el disco duro de su ordenador, con lo que la ejecución se realizará en este último.
    - Ejecutar un componente desde la fuente de donde proviene (CD-ROM o DVD, normalmente), lo que le ahorrará espacio en su disco duro.
    - No instalar un componente determinado hasta que no sea requerido por primera vez (instalación por demanda), de esta forma se evita la instalación de componentes que no se utilicen nunca y se ahorrará espacio en disco. Si en un momento dado se necesitara hacer uso de esa componente será en ese momento cuando se realizará la instalación.

- No instalar en ningún caso un componente determinado.

Dará opción también a decidir el lugar del disco en el que se realizará la instalación, el idioma en el que se desea que aparezcan los mensajes al usuario y algunas otras características como apariencias, utilización de asistentes, etc., propias de la aplicación que se está instalando.

Finalmente, e independientemente de la opción elegida, el programa de instalación pedirá al usuario que registre la copia del producto adquirido para poder así recibir soporte técnico y posibles actualizaciones.

3.4 Documentación

Al final del ciclo la documentación generada se completa con un manual de gran importancia que, aunque de alguna manera se han ido realizando durante el proyecto, ahora se formaliza para entregarlo: el manual de usuario.

Así mismo, se generará un manual de explotación, manual que definiremos en el punto 5º del presente tema.

Manual de Usuario:

Reúne la información, normas y documentación necesaria para que el usuario conozca y utilice adecuadamente la aplicación desarrollada.

Se persigue con el los siguientes objetivos:

- Correcto uso de la aplicación
- Utilidad para conseguir y detectar errores.
- Utilidad para formar a usuarios.
- Uso como guía de consulta y referencia.

En su redacción debe ser claro, conciso, correcto y completo, incluyendo los apoyos gráficos que se crean necesarios para su correcta comprensión por parte de los destinatarios.

4. MANTENIMIENTO (Opcional al tema)

El mantenimiento software es mucho más que "una corrección de errores". Se puede describir el mantenimiento describiendo las cuatro actividades que se llevan a cabo al distribuir una aplicación software:

- Mantenimiento correctivo.
- Mantenimiento adaptativo
- Mantenimiento perfectivo
- Ingeniería inversa

4.1. MANTENIMIENTO CORRECTIVO

La primera actividad de mantenimiento, es debida a que no es razonable asumir que la prueba del software haya descubierto todos los errores latentes de un gran sistema software.

El proceso que incluye el diagnóstico y la corrección de uno o más errores se denomina mantenimiento correctivo.

4.2. MANTENIMIENTO ADAPTATIVO

La segunda actividad que contribuye a la definición de mantenimiento, se produce por el rápido cambio inherente a cualquier aspecto de la informática.

La vida útil del software de aplicación, puede fácilmente superar los diez años, sobreviviendo al entorno del sistema para el que fue originalmente desarrollado.

El mantenimiento adaptativo es la actividad que se encarga de modificar el software para que interaccione adecuadamente con su entorno cambiante.

4.3. MANTENIMIENTO PERFECTIVO

La tercera actividad que se puede aplicar a la definición de mantenimiento se produce cuando un paquete de software tiene éxito.

Con respecto a esta actividad se define el mantenimiento perfectivo, el cual se encarga de mantener el software con respecto a las peticiones que hacen los usuarios sobre las nuevas posibilidades y mejoras en general.

4.4. INGENIERÍA INVERSA

La cuarta actividad de mantenimiento se da cuando se cambia el software para mejorar una futura facilidad de mantenimiento o fiabilidad, o para proporcionar una base mejor para futuras mejoras.

Para realizar esta actividad se utilizan técnicas de ingeniería inversa o reingeniería.

5. EXPLOTACIÓN DE APLICACIONES INFORMÁTICAS

Para medir la explotación de una aplicación o un sistema informático se utilizan los índices de rendimiento, que pueden ser de dos tipos:

- **Internos:** cuantifican la eficiencia de los distintos componentes del sistema. Algunos ejemplos de índices de rendimiento internos son: utilización de la CPU, solapamiento de actividades, nivel de multiprogramación y ratio de paginación.
- **Externos:** tratan con el rendimiento global del sistema. Algunos ejemplos son: tiempo de respuesta, productividad, disponibilidad, y confiabilidad.

Como anteriormente citamos, al final del ciclo, la documentación generada incluirá un manual de explotación.

Manual de explotación:

Recoge la información necesaria para poner en explotación la aplicación. Va dirigido a la instalación fundamentalmente, en cuanto a operatividad práctica, turnos, etc.

En general, recogerá información común a los primeros puntos del manual de usuarios más aquella dirigida a hacer operable el producto como disposición de archivos, periodicidades de ejecución, comunicación con el operador del sistema, etc.

5.1. OBJETIVOS DE LA EVALUACIÓN DE SISTEMAS

La determinación del rendimiento de un sistema es una necesidad en diferentes etapas de su vida útil.

Para ello se realizan tres tipos de estudios:

1. **Diseño.** Se investiga el rendimiento para obtener una configuración óptima de los componentes del sistema.
2. **Mejora y Sincronización.** En la fase de productividad es posible que el rendimiento alcance unos valores inaceptablemente bajos. Se debe intentar reducir la carga del sistema y optimizar los valores de determinados índices de rendimiento.
3. **Planificación**. Se intenta predecir el comportamiento del sistema frente a modificaciones de la carga de trabajo.

5.2. TÉCNICAS DE EVALUACIÓN

Los métodos de evaluación son aquellos sobre los cuales se obtienen los valores de los índices de rendimiento.

Las técnicas empleadas son:

- De **medida**. Mide directamente las variables del sistema, siempre teniendo en cuenta que el sistema tendrá que estar en uso. Es la más fiable pero no siempre la más adecuada.

- De **modelaje.** Se construye el modelo conceptual, sobre él existen dos posibles aproximaciones para la obtención de valores:

  - Simulación. Se reproduce su comportamiento

  - Técnicas analíticas. Se utilizan expresiones matemáticas para tomar los valores.

  Esta última técnica es aproximada y no siempre exacta. La ventaja es que no necesita la presencia física del sistema.

6. COMPARTICIÓN DE DATOS

La instalación y explotación de aplicaciones informáticas sobre sistemas que comparten datos, implica tener en cuenta aspectos no considerados hasta ahora. Dichos aspectos van a variar dependiendo del tipo de sistema que se trate:

- Sistema multiusuario: donde pueden trabajar varios usuarios simultáneamente.
- Sistema multiprogramación o multitarea: un sistema puede ejecutar varias aplicaciones simultáneamente

6.1. SISTEMAS MULTIUSUARIO

En un sistema multiusuario, casi siempre existe la necesidad de permitir a los usuarios compartir archivos. Esta necesidad presenta dos cuestiones importantes:

- Derechos de acceso
- Gestión de los accesos simultáneos.

6.1.1. Derechos de acceso

El **sistema de archivos** debe ofrecer una herramienta flexible para permitir la compartición general de archivos entre los usuarios, así como, un conjunto de opciones de forma que se pueda controlar la manera en que se accede a cada a archivo en particular.

Entre los **derechos de acceso** que pueden asignarse a un usuario particular para un archivo específico son:

1. Ninguno, el usuario no conoce la existencia del archivo.
2. Conocimiento, el usuario conoce su existencia y su propietario.
3. Ejecución, el usuario puede cargar y ejecutar un programa pero no copiarlo.
4. Lectura, el usuario puede leer el archivo para cualquier propósito (incluyendo copiar y ejecución).
5. Adición, el usuario puede añadir datos (generalmente al final del archivo), pero no modificar o borrar el contenido del mismo.
6. Actualización, permite modificar, borrar y añadir datos al archivo.
7. Cambio de protección, el usuario puede cambiar los derechos de accesos otorgados a otros usuarios.
8. Borrado, el usuario puede borrar el archivo del sistema de archivos.

6.1.2. Accesos simultáneos

Cuando se otorga acceso para añadir o actualizar a más de un usuario, se debe de hacer cumplir una disciplina, que puede consistir en:

- **Bloquear el archivo** entero cuando se vaya a actualizar
- **Bloquear los registros** individuales durante la actualización.

La segunda técnica es más difícil de implementar, pero también es la más eficiente.

6.2. MULTIPROGRAMACIÓN

La introducción de la multiprogramación originó la posibilidad de compartir recursos entre los usuarios.

La compartición compromete al procesador, la memoria, los dispositivos de E/S, los programas y los datos.

La capacidad de compartir recursos introduce, la necesidad de **protección.** La protección se puede clasificar en los siguientes niveles:

1. Ninguna protección.
2. Aislamiento, este enfoque implica que cada proceso opera separadamente, es decir, sin compartición ni comunicación.
3. Compartir todo o nada, el propietario de los datos decide si son públicos o privados, es decir, si puede acceder cualquier usuario o proceso o si sólo podrán acceder los procesos del propietario.
4. Compartir por limitación del acceso, es el sistema operativo que comprueba para cada acceso si se trata de un acceso permitido.
5. Compartir por capacidades dinámicas, amplía el concepto de control de acceso, incorporando la creación dinámica de derechos de compartición para los objetos.
6. Uso limitado de un objeto, esta forma de protección limita no sólo el acceso a un objeto, sino también la utilidad a que se puede dedicar dicho objeto. Así, se podrá permitir a un usuario el acceso a un documento, pero no su impresión.

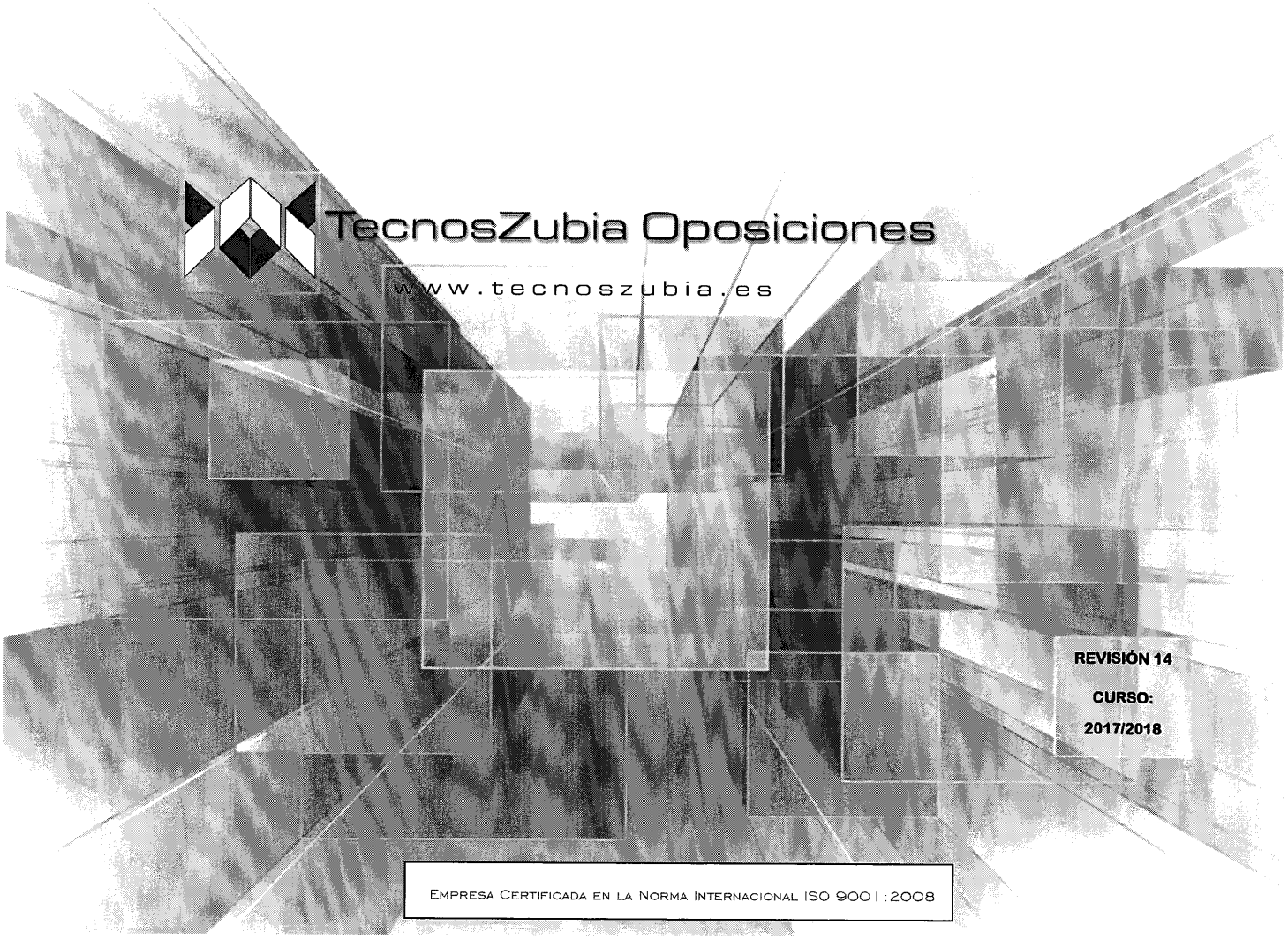
7. BIBLIOGRAFÍA

- Rocandio Pablo, F.J.; Tecnologías de la información. Edit. McGraw-Hill.
- De Miguel Anasagasti, Pedro; Fundamentos de los computadores. Edit. Paraninfo.

SISTEMAS Y APLICACIONES INFORMÁTICAS

TEMA 50 (S.A.I.):

Calidad y documentación en entornos gráficos.



TecnosZubia Oposiciones

www.tecnoszubia.es

REVISIÓN 14

CURSO: 2017/2018

TEMA 50 "Calidad y documentación en entornos gráficos."

0 INTRODUCCIÓN

La culminación de las actividades de desarrollo de un nuevo producto de software en entornos gráficos exige, especialmente hoy en día, una sistemática que permita poner a prueba el sistema concebido, de forma que se asegure que se encuentra listo para su distribución.

La revolución que supuso en el desarrollo de software el hecho de seguir metodologías y herramientas propias de otros ámbitos de ingeniería hizo también que se exigiesen políticas de gestión de la calidad del software, al igual que dichas políticas existían en otras ingenierías.

La gestión de la calidad en ingeniería requiere una cuantificación o métrica. El reto consistirá, pues, en aplicar a un producto intangible como es el software una serie de mediciones para caracterizarlo.

La creación de aplicaciones en entornos gráficos y de los mismos interfaces gráficos conlleva un alto grado de complejidad en el desarrollo de software y en la integración de sistemas. Los proyectos involucrados en tal creación necesitan de la documentación del software como indicador clave en el control de la calidad.

En este tema, pues, expondremos los conceptos y técnicas relacionados con la calidad y la documentación del software en entornos gráficos, así como su cuantificación.

1 CALIDAD EN ENTORNOS GRÁFICOS.

1.1. Concepto de calidad de software

Podemos definir "Calidad del Software" como la existencia de concordancia entre los requisitos funcionales y de rendimiento explícitamente establecidos, y los estándares de desarrollo explícitamente documentados, además de las características implícitas que se espera de todo software desarrollado adecuadamente.

Los requisitos del software son los fundamentos desde los que se mide la calidad. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se desarrolla el software. Existe un conjunto de requisitos implícitos que a menudo no se mencionan, como una interfaz amigable, eficiencia, etc.

1.2. Concepto de entorno gráfico

Cuando hablamos de entorno gráfico nos referimos a la cara visible de los programas tal y como se presenta a los usuarios para que interactúen con la máquina. La interfaz gráfica implica la presencia de una pantalla constituida por una serie de menús e iconos que representan las opciones que el usuario puede tomar dentro del sistema.

El primer entorno moderno de escritorio que se comercializó fue desarrollado por Xerox en los años ochenta. Actualmente el entorno más conocido es el ofrecido por la familia Windows aunque existen otros como los de Macintosh (Classic y Cocoa) y de código abierto (o software libre) como GNOME, KDE, etc.

1.3. Características de un software de calidad

Para que un software pueda cumplir con unos niveles de calidad aceptable, deberá cumplir con una serie de requisitos o características:

- **Fiabilidad:** grado en que un programa satisface sus especificaciones y consigue los objetivos exigidos.
- **Flexibilidad:** El esfuerzo requerido para modificar un programa operativo.
- **Corrección:** El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente.
- **Portabilidad:** facilidad para transferir el programa desde un hardware y/o un entorno de sistema de software a otro.
- **Eficiencia:** cantidad de recursos de computadora y código requeridos por un programa para llevar a cabo sus funciones.
- **Reusabilidad:** El grado en que un programa (o partes de un programa) se puede reutilizar en otras aplicaciones.
- **Integridad:** El grado en que puede controlarse el acceso al software o a los datos por personal no autorizado.
- **Ingeniería humana:** grado en que el software facilita la funcionalidad y el rendimiento asignados al elemento humano.
- **Facilidad de prueba:** esfuerzo requerido para probar un programa de forma que se asegure que realiza su función requerida.
- **Facilidad de uso:** esfuerzo requerido para aprender y trabajar con un programa, así como preparar la entrada e interpretar la salida de un programa.
- **Facilidad de mantenimiento:** esfuerzo requerido para localizar y reparar un error en un programa.
- **Facilidad de interoperación:** El esfuerzo requerido para acoplar un sistema a otro.

Sería deseable maximizar estas características, pero este objetivo es difícil de alcanzar porque algunas de estas características son antagonistas (si una

aumenta otra disminuye), en general la alta calidad se consigue con un alto coste y porque es difícil definir una métrica simple que se pueda usar para medir una característica particular.

1.4 Métricas de la calidad del software.

Para poder establecer la calidad de un software, el primer paso a realizar es definir una métrica que permita cuantificar y comparar algunas de las características antes descritas.

Existen múltiples métricas, entre las que destacamos:

1.4.1. Métrica de Halstead

El método consiste en determinar la productividad por el número y por los tipos de palabras usadas en el programa. Sirve para predecir la productividad de desarrollo y mantenimiento y también para indicar la complejidad.

Esta métrica, por tanto, requiere disponer del código del programa, que se considera una colección de palabras que pueden ser clasificadas en operadores y operandos.

1.4.2. Medida de complejidad de McCabe.

La métrica está basada en la teoría matemática de grafos. Un programa se ve como un grafo con un único punto de entrada y un único punto de salida. La complejidad ciclomática viene dada por:

$$V(G) = e - n + 2p$$

Donde

e = número de aristas en el grafo.

n = número de nodos en el grafo

p = número de componentes conectados en el grafo.

1.5 Pruebas

La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Es un elemento crítico para la garantía de la calidad del software. La prueba es un proceso que se enfoca sobre:

- La lógica interna del software, y
- Las funciones externas.

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a una construcción correcta del software.

Existen 2 métodos generales de diseño de pruebas:

- Pruebas de Caja Blanca, que permiten examinar la estructura interna de un programa.
- Pruebas de Caja Negra, donde los casos de prueba se diseñan considerando exclusivamente las entradas y las salidas del sistema, sin preocuparse de la estructura interna del mismo.

Las diferentes fases que el proceso de pruebas ha de seguir en el desarrollo de un sistema software pueden verse en la siguiente tabla:

NIVEL	FASE DE PRUEBA
Codificación	P. de Unidad
Diseño	P. de Integración
Requisitos	P. de Validación
Sistema	P. de Sistema

1.5.1. Pruebas de unidad

Como hemos indicado en la tabla anterior, estas pruebas se llevan a cabo en la fase de codificación del software y están destinadas a probar cada una de las unidades del sistema, esto es, se centran en la verificación de la menor unidad en el diseño del software: el módulo.

1.5.2. Pruebas de integración

Técnica sistemática que nos permite construir el programa completo a partir de módulos probados. También se detectan errores de integración.

El objetivo es tomar los módulos probados en la prueba de unidad y construir con ellos una estructura de programa que esté de acuerdo con lo que dicta el diseño.

1.5.3. Pruebas de validación

Estas pruebas están destinadas a comprobar que el software funciona de acuerdo con lo exigido por el cliente, esto es, si se ajusta a los requisitos. En este punto, se llevan a cabo lo que se llaman pruebas de caja negra.

Hace falta, al igual que en las etapas anteriores, un diseño cuidadoso de pruebas que demuestren la validez del software.

Las pruebas se llevan a cabo entre el equipo de prueba y los potenciales usuarios del producto. Para ello, se instala el software de forma provisional y se solicita a los usuarios que anoten todos los errores que aparezcan. Hay dos tipos:

- **Prueba Alfa:** Se realiza en el lugar de desarrollo del producto. El cliente realiza la prueba. El que desarrolla el software va registrando los problemas que aparecen en el proceso de prueba.
- **Prueba Beta:** Se realiza en el lugar donde se instala el sistema completo, es realizado y conducido únicamente por el cliente, él mismo va anotando los problemas que va encontrando.

1.5.4. Pruebas del sistema

En esta fase, el sistema de software construido debe quedar en perfecto estado de funcionamiento. Para ello se realizan, entre otras, las siguientes tareas:

- Se adaptan y se ultiman detalles acerca del hardware sobre el que, definitivamente, se va a implantar el sistema.
- Se fuerzan errores en el sistema para comprobar que se recuperan (o no).
- Se verifican los mecanismos de protección incorporados al sistema (clave de acceso, seguridad en los ficheros, mecanismos "anti-pirateo", etc.).

- Se llevan a cabo pruebas que pongan de manifiesto cuál es el nivel de rendimiento del sistema.

- Se llevan a cabo pruebas de resistencia cuya finalidad es enfrentar al sistema con situaciones anormales.

1.6 Factores de calidad en entornos gráficos

La calidad del software en numerosas ocasiones tiene su escaparate a través del interfaz gráfico para el usuario final. Apareciendo reflejados diversos factores que intervienen en el incremento de calidad de un software a través del interfaz gráfico de usuario. Los factores más importantes a tener en cuenta son:

- **Usabilidad:** Se trata de una serie de técnicas que ayudan a los usuarios a realizar tareas en entornos gráficos utilizando un ordenador. La usabilidad ayuda a que esas tareas se realicen de una forma sencilla analizando el comportamiento humano, y los pasos necesarios para ejecutar la tarea de una forma eficaz.

- **Rapidez y fiabilidad:** Una aplicación o una web por ejemplo, de cara al usuario debe ser rápida y fiable. De poco sirve crear una web muy compleja si tarda mucho en cargar todos sus elementos y merma la paciencia del usuario.

- **Seguridad:** Este factor es determinante en la confianza del usuario final.

- **Simplicidad y optimización:** Se trata de diseñar software sencillo de entender y de utilizar por el usuario. Para generar en él la sensación de comodidad con el entorno, así como de aprovechamiento de las utilidades.

1.7 Normalización en el diseño de Interfaces Gráficas

Las normas sobre calidad del software se recogen en la serie ISO 9000. Dentro de éstas, la norma más relevante en la **Human-computer interaction (HCI)** es la **ISO 9241**, que trata los requisitos para trabajar con pantallas (*Visual display terminals, VDTs*) y la reciente **ISO/TR 16982**, sobre ergonomía y usabilidad en el diseño. Se ocupa de la colocación de los distintos dispositivos del ordenador y de la postura que el usuario adopta para su uso, del diálogo –la interacción- entre el hombre y la máquina y de los aspectos de software relacionados con el diseño en pantalla.

Además, existen **guías de estilo** o conjuntos de directrices producidas por empresas fabricantes de ordenadores y de desarrollo de software que cubren características de detalle de bajo nivel. Su objetivo es producir grupos de productos (paquetes) con el mismo aspecto de modo que los usuarios puedan utilizar un programa u otro sin dificultades de manejo.

2 DOCUMENTACIÓN EN ENTORNOS GRÁFICOS

La documentación consiste en un conjunto de técnicas y documentos que explican las características internas de los programas. El estilo de la documentación conlleva una filosofía mezcla de simplicidad y de claridad.

2.1 Características

Las características de la documentación del software son:

- Mejora la legibilidad del programa, ya que explica lo que hace en cada momento.
- Facilita la labor del creador del programa, porque pasado un tiempo lo más seguro es que no entienda las características internas del programa, si no está bien documentado.
- La documentación se utiliza en todas las fases del ciclo de vida y es un factor importante para medir el avance y para el control de calidad.
- La documentación es especialmente importante para el mantenimiento del programa.

2.2 Técnicas de documentación

Existen tres grupos de personas que necesitan conocer la documentación del software: programadores, operadores y usuarios. Los requisitos necesarios para cada uno de ellos suelen ser diferentes, en función de las misiones de cada grupo:

- Programador manual de mantenimiento del programa
- Operador manual del operador
- Usuario manual de usuario

En entornos interactivos, las misiones del usuario y del operador suelen ser las mismas. Así la documentación del programa se puede concretar en manual de usuario y manual de mantenimiento.

2.2.1. Manual de usuario

La documentación de un paquete de software suele producirse con dos propósitos:

- Explicar las funciones del software y describir el modo de utilizarlas (documentación de usuario) porque está diseñada para ser leída por el usuario del programa.
- Describir el software en sí para poder mantener el sistema en una etapa posterior de su ciclo de vida (documentación del sistema o de mantenimiento).

La documentación de usuario es un instrumento comercial importante. Una buena documentación de usuario hará el programa más accesible. Esta documentación adopta la forma de un manual que presenta:

- Una introducción de las funciones más utilizadas del software.
- Una sección que explica cómo instalar el programa.
- Una sección de referencia que describe los detalles de cada función del software.

Es frecuente que el manual se edite en forma de libro, aunque cada vez es más frecuente incluirlo además, o en lugar, del libro en el propio programa y suele denominarse **manual de ayuda en línea.**

El manual de usuario debe cubrir al menos los siguientes puntos:

- Órdenes necesarias para instalar el programa y comenzar su funcionamiento.

- Nombres de archivos externos a los que el programa accede.
- Formato de todos los mensajes de error o informes.
- Opciones en el funcionamiento del programa.
- Descripción detallada de la función realizada por el programa.
- Descripción detallada, preferiblemente con ejemplos, de cualquier salida producida por el programa.

2.2.2. Manual de mantenimiento

La **documentación del sistema** o **manual de mantenimiento** es por naturaleza, más técnica que la de usuario. Abarca todo el ciclo de vida de desarrollo del software, incluidas las especificaciones originales del sistema y aquellas con las que se verificó el sistema, los diagramas de flujo de datos (DFD), diagramas entidad-relación (DER), diccionario de datos y diagramas o cartas de estructura que representan la estructura modular del sistema.

Se divide en dos categorías: documentación interna y documentación externa.

- **Documentación interna:** Esta documentación cubre los aspectos del programa relativos a la sintaxis del lenguaje. Esta documentación puede estar contenida en los comentarios del programa. Algunos tópicos a considerar son:
  - Cabecera de programa (nombre del programador, fecha de la versión actual, breve descripción del programa).
  - Nombre significativos para describir identificadores.
  - Comentarios relativos a la función del programa, así como de los módulos que componen el programa.

- Claridad de estilo y formato (una sentencia por línea, indentación), líneas en blanco para separar módulos, etc.

- **Documentación externa:** Documentación ajena al programa fuente. Frecuentemente, se emplean herramientas de autogeneración de documentación del código como: javadoc, Doxygen, etc. Las cuales extraen los comentarios del código fuente y crean manuales de referencia en documentos de texto o archivos HTML. La documentación externa debe incluir:

  - Listado actual del programa fuente, mapas de memoria, referencias cruzadas, etc.

  - Especificación del programa: documentar el propósito y modo de funcionamiento del programa.

  - Diagrama de estructura que representa la organización jerárquica de los módulos que comprende el programa.

  - Explicaciones de fórmulas complejas.

  - Especificación de los datos a procesar: archivos externos incluyendo al formato de las estructuras de los registros, campos, etc.

  - Formatos de pantallas utilizados para interactuar con los usuarios.

  - Cualquier indicación especial que pueda servir a los programadores que deben mantener el programa.

2.3 Reglas de documentación

Un programa bien documentado es aquél que otras personas pueden leer, usar y modificar. Existen muchos estilos aceptables de documentación y, con frecuencia, los temas a incluir dependerán del programa específico. No obstante, señalamos a continuación algunas características esenciales comunes a cualquier documentación de un programa:

1. Un comentario de cabecera para el programa que incluye:
    - Descripción de programa: propósito.
    - Autor y fecha.
    - Descripción de la entrada y salida del programa.
    - Descripción de cómo utilizar el programa.
    - Hipótesis sobre tipos de datos esperados.
    - Breve descripción de los algoritmos globales y estructuras de datos.
    - Descripción de las variables importantes.
2. Comentarios breves en cada módulo similares a la cabecera del programa y que contenga información adecuada a ese módulo, incluyendo en su caso precondiciones y postcondiciones. Describir las entradas y cómo las salidas se relacionan con las entradas.
3. Escribir comentarios inteligentes en el cuerpo de cada módulo que expliquen partes importantes y confusas del programa.
4. Describir claramente y con precisión los modelos de datos fundamentales y las estructuras de datos seleccionadas para representarlas así como las operaciones realizadas para cada procedimiento.

Aunque existe la tendencia entre los programadores y sobre todo entre los principiantes a documentar los programas como última etapa, esto no es buena práctica, lo idóneo es documentar el programa a medida que se desarrolla. La tarea de escribir un programa grande, se puede extender por períodos de semanas o incluso meses. Esto le ha de llevar a la consideración de que lo que resulta evidente ahora puede no serlo de aquí a dos meses; por esta causa, documentar a medida que se progresa en el programa es una regla de oro para una programación eficiente.

Medida de la calidad en la documentación

Como ocurre con la cuantificación de la calidad del software a través de métricas y pruebas, la de un documento también podría realizarse si se consigue identificar un conjunto suficientemente representativo de variables de medida para ello. En este sentido hay que diferenciar, por una parte, la medida de la calidad del texto incluido en un documento, la calidad de la interfaz del documento con el usuario, la calidad de la estructura del documento, sobre todo en documentos hipertexto o hipermedia, y, por otro lado, la calidad de los diagramas o modelos que aparecen en ellos como consecuencia de haber aplicado alguna metodología de desarrollo de software. En este último caso, el establecimiento de variables de medida es más sencillo, pues vienen impuestas por la propia técnica de modelado, existiendo métricas tanto para modelos estructurados como orientados a objetos.

Más complicada es la cuantificación de la calidad global de un documento, que normalmente se refiere a la medida de su comprensibilidad. Los principales métodos de medida pueden agruparse en las siguientes categorías establecidas por Lehner [1993]:

- **Métodos basados en las pruebas de CLOZE**: Su nombre proviene de *closure construct* y hace referencia a la característica humana que

permite completar caracteres incompletos según el contexto. El principal método de este tipo es el "*procedimiento de Taylor*", utilizado inicialmente en el ámbito periodístico (años cincuenta). También ha sido utilizado en la documentación de proyectos de software [Hall y Zweben, 1986].

- **Métodos de formalización de la comprensibilidad:** Se caracterizan por determinar la comprensibilidad y legibilidad de los textos cuantificando formalmente atributos léxicos y sintácticos. Son los métodos más científicos, que obtienen índices de comprensibilidad a partir de diferentes fórmulas matemáticas. Algunos índices propuestos por estos métodos son *Fog index, Steiwer index, etc.*

- **Métodos de ordenación/estructuración de la información:** Con estos métodos se trataría de "medir" la adecuada disposición de los contenidos según un orden secuencial, desde aspectos generales hasta aspectos específicos. Este tema ha estado bastante olvidado hasta la aparición de los documentos hipertexto e hipermedia, cuyas enormes posibilidades de navegación por su contenido ha planteado la necesidad de establecer métricas para evaluar la calidad de los enlaces hipermediales [Johnson, 1995]. Autores como Botafogo, Rivlin y Shneiderman [1992] han propuesto un método formal de cuantificación de la calidad basado en el análisis de la estructura jerárquica de los hipertextos, midiendo, entre otros, la *compactación* (lo intrincado de las conexiones) y la *disposición* (el grado de organización, que se refiere al número de nodos que deben ser leídos antes que otros) del documento.

- **Métodos basados en el medio de presentación:** Se basan en cuantificar la calidad de la forma de presentar la información al lector. Esto tiene una gran importancia en el caso de la documentación hipermedia, cuya lectura se realiza a través de un interfaz de usuario cuya calidad debería poder valorarse, ya que influirá en la atención que se prestará en la lectura y, por ende, en la comprensión del contenido. En este sentido existen autores como [Hammond, 1993] y [Thüring , 1995], que han analizado los parámetros que influyen en la comprensión

de un hiperdocumento teniendo en cuenta el proceso cognitivo que se produce en el lector cuando lo utiliza, estos parámetros son el *control*, la *participación* y la *síntesis*.

- **Métodos intuitivos basados en la experiencia:** Son métodos sin justificación científica, pero que han adquirido un cierto nivel de aceptación, como el de Reiners, que promedia la opinión de un cierto número de lectores respecto a la comprensión individual de palabras (o frases), verbos activos, nombres personales y sustantivos abstractos.

- **Métodos basados en impresiones subjetivas:** Consiste en cuantificar las opiniones subjetivas de expertos en forma de grado de satisfacción con respecto a atributos como simplicidad, organización, brevedad, estimulación, etc.

- **Métodos basados en la relación texto-lector:** Consideran la comprensión como una interrelación ente el lector y el texto. El principal método de este tipo es el propuesto por Groeben, que evalúa la comprensibilidad de un texto a través de cuatro dimensiones: la estructura cognitiva del texto, la simplicidad lingüística y la brevedad semántica, la redundancia, y la conflictividad en la estimulación cognitiva (incongruencias, incoherencias, complejidad).

Aunque todo lo anterior es perfectamente aplicable al caso particular de la valoración de la calidad de la documentación de los proyectos de desarrollo de software, especial mención en el campo del software merecen una serie de indicadores de calidad de los documentos (*DQI: Document Quality Indicators*).

Hay otros autores, como [Torkzadeh y Doll, 1993], que se han especializado exclusivamente en la cuantificación de la calidad de los manuales de usuario, por la importancia comercial de estos documentos, ya que de ellos va a depender en gran medida el grado de satisfacción de los usuarios con la aplicación informática que han adquirido.

Además de las métricas anteriores, la documentación del software puede considerarse un proceso de ingeniería, y como tal será susceptible de ser medida como proceso, además de como producto, utilizando métricas similares a las empleadas en otras ingenierías.

3 BIBLIOGRAFÍA

- Pressman, R. S. ***Ingeniería del Software. Un enfoque práctico***, *3ª edición*. Ed. McGraw-Hill, 2000.
- Sommerville, I.: ***Ingeniería de Software***. *6ª* Edición. Addison-Wesley Iberoamericana, 2002.
- Piattini, M.y otros: ***Análisis y diseño detallado de Aplicaciones Informáticas de Gestión.*** RA-MA, 1996.
- Cerrada, J. A.: ***Introducción a la Ingeniería del Software***. *1ª Edición de 2000. Editorial Centro de Estudios Ramón Areces, S.A.*
- Lehner: ***Quality control in software documentation: Measurement of text comprehensibility*** *1993. Information and Management.*