TEMA 9: MANTENIMIENTO.

9.1.- INTRODUCCIÓN:

La creencia habitual de un equipo de trabajo de que su tarea ha finalizado cuando instala y pone en funcionamiento el software en las instalaciones del cliente no puede ser más errónea. Un producto software envuelve muchos aspectos y características que provocan que sea totalmente necesario supervisar su funcionamiento correcto durante un tiempo después de la entrega del mismo. Ante la dificultad que entraña garantizar el comportamiento correcto del programa en circunstancias no previstas, los test de aceptación del producto incluyen pruebas a largo plazo del software (a petición del cliente). A esta fase de supervisión se le denomina fase de **operación**. Sólo cuando termina esta fase el cliente acepta definitivamente el producto, que había sido aceptado provisionalmente al ser entregado (fase de transferencia).

Más tarde, es posible que el software necesite ser modificado, ya sea consecuencia de la detección de errores o bien ante nuevas exigencias y/o necesidades del usuario del sistema. A esta fase se le conoce como fase de **mantenimiento**. Es importante reseñar que durante estas fases de operación y mantenimiento (OM) se debe generar y actualizar el denominado **documento de historia del proyecto (DHP)**; documento que incluye todos los errores (y sus correcciones) y/o modificaciones realizadas en el producto. Este documento es de gran ayuda para poder calcular y analizar la fiabilidad del sistema software a la vez que evaluar el rendimiento del equipo de trabajo.

Un proyecto informático se desarrolla para la consecución de un Sistema hardware, software o combinación de ambos. La responsabilidad principal se centra en el software, ya que el coste de mantenimiento hardware cada vez es menos importante. Las empresas de hardware basadas en el mantenimiento han sufrido graves problemas de supervivencia, ya que este tipo de mantenimiento no supera el 8% del mantenimiento total de un sistema informático. Por este motivo, nos centraremos principalmente en el estudio del mantenimiento del software.

9.2.- INGENIERÍA Y MANTENIMIENTO DEL SOFTWARE:

No es el mismo tipo de mantenimiento el del software que el de hardware, como primera aproximación al mantenimiento del software lo definiremos como el conjunto de medidas que hay que tomar para que el sistema siga trabajando correctamente.

Entre las características sobresalientes del mantenimiento del software destacan:

- El software no envejece.
- El mantenimiento del software supone adaptar el paquete o sistema objeto del mismo a nuevas situaciones como:
    - Cambio de hardware.
    - Cambio de software de base (S.O.).
- Todo sistema software conlleva mejoras o añadidos indefinidamente.

Al cerrar todo proyecto se debe considerar y preveer las normas del mantenimiento del sistema (tanto en connotaciones hardware como software).

9.2.1.- EL CICLO DE VIDA Y EL MANTENIMIENTO DEL SOFTWARE:

Las fases principales en el *Ciclo de Vida del Software* son:

- Análisis y Definición de Requisitos.
- Especificación.
- Diseño.
- Programación (escritura del código).
- Prueba e instalación.
- Operación y mantenimiento.

Según ANSI-IEEE, el mantenimiento del software es la modificación de un producto software después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de entorno.

9.2.2.- COSTES DEL MANTENIMIENTO:

El coste del mantenimiento de un producto software a lo largo de su vida útil es superior al doble de los costes de su desarrollo.

Fechas	% Mantenimiento
Años 70	35% - 40%
1980 – 1984	55%
1985 – 1989	75%
Años 90	80% - 90%

Evolución de los costes del mantenimiento

Por norma general, el porcentaje de recursos necesarios en el mantenimiento se incrementa a medida que se genera más software. A esta situación se le conoce como *Barrera de Mantenimiento*.

Las causas a las que se debe este incremento de trabajo de mantenimiento son:

1) Gran cantidad de software antiguo (más de 10 años); aún siendo construidos con las mejores técnicas de diseño y codificación del momento (rara vez), su creación se produjo con restricciones de tamaño y espacio de almacenamiento y con herramientas desfasadas tecnológicamente.
2) Los programas sufren migraciones continuas de plataformas o SSOO.
3) El software ha experimentado modificaciones, correcciones, mejoras y adaptaciones a nuevas necesidades de los usuarios. Además, estos cambios se realizaron sin técnicas de reingeniería o ingeniería inversa, dando como resultado sistemas que funcionan con baja calidad (mal diseño de estructuras de datos, mala codificación, lógica defectuosa y escasa documentación).

Como consecuencia de estos grandes costes, es que el coste relativo de reparar un error aumenta considerablemente en las últimas fases del ciclo de vida del software.

Las razones por las que es menos costoso reparar defectos en las primeras fases del ciclo de vida software son:

- Es más sencillo cambiar la documentación que modificar el código.
- Un cambio en las fases posteriores puede repercutir en cambiar toda la documentación de las fases anteriores.
- Es más sencillo detectar un error en la fase en la que se ha introducido que detectarlo y repararlo en fases posteriores.

- Un defecto se puede ocultar en la inexistencia o falta de actualización de los documentos de especificación o diseño.

Existen otra serie de costes intangibles del mantenimiento del software, que son:

- Oportunidades de desarrollo que se han de posponer o que se pierden debido a los recursos dedicados a las tareas de mantenimiento.
- Insatisfacción del cliente cuando no se le satisface en un tiempo debido una solicitud de reparación o modificación.
- Los cambios en el software durante el mantenimiento también introducen errores ocultos.
- Perjuicios en otros proyectos de desarrollo cuando la plantilla tiene que dejarlos o posponerlos debido a una solicitud de mantenimiento.

En conclusión, la productividad en LDC (líneas de código) por persona y mes durante el proceso de mantenimiento puede llegar a ser 40 veces inferior con respecto al proceso de desarrollo.

9.2.3.- TIPOS DE MANTENIMIENTO:

Existen 4 tipos de mantenimiento:

- Correctivo.
- Adaptativo.
- Perfectivo.
- Preventivo.

9.2.3.1.- Mantenimiento correctivo:

Tiene por objetivo localizar y eliminar los posibles defectos de los programas. Un defecto en un sistema es una característica del sistema con el potencial de provocar un fallo. Un fallo se produce cuando el comportamiento de un sistema difiere con respecto al comportamiento definido en la especificación.

Los fallos en un sistema software pueden ser:

- Procesamiento (salidas incorrectas de un programa).
- Rendimiento (tiempo de respuesta demasiado alto).
- Programación (inconsistencias en el diseño).
- Documentación (inconsistencias entre la funcionalidad de un programa y el manual de usuario).

9.2.3.2.- Mantenimiento adaptativo:

Consiste en la modificación de un programa debido a cambios en el entorno (hardware o software) en el que se ejecuta. Desde cambios en el sistema operativo, pasando por cambios en la arquitectura física del sistema informático, hasta en el entorno de desarrollo del software. Este tipo de mantenimiento puede ser desde un pequeño retoque hasta una reescritura de todo el código.

Los cambios en el entorno de desarrollo del software pueden ser:

- En el entorno de los datos (p.e. cambiar sistema de ficheros por BD relacional).

- En el entorno de los procesos (p.e. migración a plataforma con procesos distribuidos).

Este mantenimiento es cada vez más frecuente debido a la tendencia actual de actualización de hardware y SSOO cada poco tiempo.

9.2.3.3.- Mantenimiento perfectivo:

Conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario.

Se divide en dos:

- *Mantenimiento de Ampliación*: incorporación de nuevas funcionalidades.
- *Mantenimiento de Eficiencia*: mejora de la eficiencia de ejecución.

9.2.3.4.- Mantenimiento preventivo:

Modificación del software para mejorar las propiedades de dicho software (calidad y mantenibilidad) sin alterar sus especificaciones funcionales. Incluir sentencias que comprueben la validez de los datos de entrada, reestructuración de los programas para aumentar su legibilidad o incluir nuevos comentarios. Este tipo de mantenimiento utiliza las técnicas de ingeniería inversa y reingeniería. El mantenimiento para la reutilización especializado en mejorar la reusabilidad del software se incluye en este tipo.

9.2.4.- ACTIVIDADES DE MANTENIMIENTO:

Las actividades de mantenimiento se agrupan en tres categorías funcionales:

Comprensión del software y de los cambios a realizar (Comprender): es necesario el conocimiento a fondo de la funcionalidad, objetivos, estructura interna y requisitos del software. Alrededor del 50% de tiempo de mantenimiento se dedica a esta actividad, a consecuencia de lo cual, las herramientas CASE incorporan utilidades que automatizan este tipo de tareas aumentando de manera notable la productividad.

Modificación del software (Corregir): crear y modificar las estructuras de datos, la lógica de procesos, las interfaces y la documentación. Los programadores deben evitar los efectos laterales provocados por sus cambios. Esta actividad representa ¼ del tiempo total de mantenimiento.

Realización de pruebas (Comprobar): realizar pruebas selectivas que nos aseguren la corrección del software.

Categoría	Actividad	% Tiempo
Comprensión del software y de los cambios a realizar	Estudiar las peticiones	18%
	Estudiar la documentación	6%
	Estudiar el código	23%
Modificación del software	Modificar el código	19%
	Actualizar la documentación	6%
Realización de pruebas	Diseñar y realizar pruebas	28%

Importancia de las actividades de mantenimiento

9.3.- DIFICULTADES DEL MANTENIMIENTO:

El proceso de mantenimiento no debe deteriorar la calidad del software. ¿ Cómo debe mantenerse el software para preservar su fiabilidad ?.

9.3.1.- CÓDIGO HEREDADO:

La mayor parte del software en la actualidad está formado por código antiguo "heredado" (legacy code); esto es, código desarrollado hace tiempo, con técnicas y herramientas en desuso y, para más INRI, por personas que actualmente no se encargan de su mantenimiento. Además puede que incluso este código haya pasado varias actividades de mantenimiento; y por otra parte, el volver a reescribirlo no compensa por la carga financiera que supuso y la necesidad de su amortización.

Leyes del Mantenimiento del Software:

- **Continuidad del cambio.-** Un programa utilizado en un entorno del mundo real debe cambiar si no quiere dejar de ser usado. Esto se debe a que surgen nuevas funcionalidades, nuevo hardware puede permitir mejoras en el software, se corrigen defectos, se instala en otro sistema operativo/máquina o el software necesita ser más eficiente.
- **Incremento de la Complejidad.**- Cuando se realizan cambios en un programa la estructura se hace más compleja si no se utilizan técnicas de ingeniería del software.
- **Evolución del programa.-** Es un proceso autorregulado. Se mantienen las tendencias e invariantes de las propiedades del programa.
- **Conservación de la Estabilidad Organizacional.-** La carga que supone el desarrollo de un programa es aproximadamente constante e independiente de los recursos dedicados a lo largo del tiempo de vida del mismo.
- **Conservación de la Familiaridad.**- El incremento en el número de cambios introducidos con cada versión (release) es aproximadamente constante.

9.3.2.- PROBLEMAS DEL MANTENIMIENTO:

a) Es habitual realizar el mantenimiento de forma *ad hoc* en un estilo libre del programador. Esto es debido a que no existen o son poco conocidos los métodos, técnicas y herramientas que proporcionan soluciones globales al problema del mantenimiento.
b) Después de cada cambio los programas tienden a ser menos estructurados. Como consecuencia se produce una documentación desfasada, código que no cumple los estándares, incremento en el tiempo de comprensión de los programas o incremento de los efectos secundarios de los cambios.
c) Los sistemas que son mantenidos son cada vez más difíciles de cambiar.
d) Los usuarios participan poco en el desarrollo del software, con el riesgo de que no satisfaga sus necesidades y aumenten los esfuerzos en el mantenimiento.
e) Problemas de gestión. Existe una visión de que el trabajo de mantenimiento es de una escala inferior al trabajo de desarrollo de software. Se realiza

mantenimiento precipitado, no documentado adecuadamente y poco integrado en el código existente.

9.3.3.- EFECTOS SECUNDARIOS DEL MANTENIMIENTO:

En el mantenimiento del software existe el riesgo del llamado efecto bola de nieve; que consiste en que los cambios introducidos por una petición de mantenimiento conllevan efectos secundarios que implican futuras peticiones de mantenimiento.

9.3.3.1.- Efectos secundarios sobre el código:

1.- Cambios en el diseño que suponen muchos cambios en el código.
2.- Eliminación o modificación de un subprograma.
3.- Eliminación o modificación de una etiqueta.
4.- Eliminación o modificación de un identificador.
5.- Cambios para mejorar el rendimiento.
6.- Modificación de la apertura/cierre de ficheros.
7.- Modificación de operaciones lógicas.

9.3.3.2.- Efectos secundarios sobre los datos:

1.- Redefinición de constantes locales o globales.
2.- Modificación de los formatos de registros o archivos.
3.- Cambio en el tamaño de una matriz u otras estructuras similares.
4.- Modificación de la definición de variables globales.
5.- Reinicialización de indicadores de control o punteros.
6.- Cambios en los argumentos de los subprogramas. Es importante una correcta documentación de los datos.

9.3.3.3.- Efectos secundarios sobre la documentación:

1.- Modificar el formato de las entradas interactivas.
2.- Nuevos mensajes de error no documentados.
3.- Tablas o índices no actualizados.
4.- Texto no actualizado correctamente.

9.4.- SOLUCIONES AL PROBLEMA DEL MANTENIMIENTO:

9.4.1.- SOLUCIONES DE GESTIÓN:

Los gestores seniors de las organizaciones deben ser conscientes de:

1.- Importancia de las tecnologías de la información para la organización.
2.- El software es activo corporativo y puede suponer una venta competitiva.

Deben centrar las soluciones en dos aspectos: recursos y calidad.

9.4.1.1.- Recursos dedicados al mantenimiento:

El recurso clave es el humano. Lo habitual es que esta tarea sea asignada a personal nuevo recién incorporado a la organización, sin experiencia en el uso de las técnicas de ingeniería del software y sin conocimiento del programa; y como consecuencia de ello, raramente consiguen encontrar y corregir defectos o realizar

modificaciones. Por lo tanto una mejora indispensable es constituir un grupo de programadores dedicados al mantenimiento de código antiguo.

9.4.1.2.- Gestión de la calidad:

Para resolver el problema a largo plazo es necesario mejorar la calidad del proceso en su conjunto.

Técnicas de gestión de la calidad del software:

- Uso de técnicas estándares para descomponer el software en entidades funcionales.
- Uso de estándares de documentación del software.
- Diseño paso a paso en cada nivel de descomposición del software.
- Uso de código estructurado.
- Definición a priori de todas las interfaces y estructuras de datos antes del diseño.
- Uso de métricas de producto (miden atributos del producto software) y métricas de procesos (evalúan la calidad del proceso).
- Uso de mejores herramientas y entornos de desarrollo de software.

9.4.1.3.- Gestión estructurada del mantenimiento:

Si el mantenimiento no es estructurado se sufren las consecuencias: dolorosa evaluación del código, complicada comprensión del sistema por la pobre documentación interna, dificultad para descubrir las consecuencias de los cambios en el código y la imposibilidad de realizar pruebas de regresión (repetición de pruebas anteriores) debido a la inexistencia de registros de pruebas.

Sugerencias para mantener código heredado:

- Obtener la máxima información sobre el programa antes de que surja el mantenimiento.
- Conocer, entender y, en caso de no existir, dibujar el flujo de control del programa.
- Evaluar la documentación.
- Añadir comentarios al código para ayudar a su comprensión posterior.
- Usar las ayudas de los compiladores.
- Respetar el estilo y formato al realizar cambios.
- Señalar las instrucciones del código cambiadas.
- Guardar copia de seguridad antes de eliminar código.
- Usar variables propias para evitar efectos secundarios.
- Mantener un registro completo de las actividades de mantenimiento.
- Añadir comprobación de errores.

Es necesario realizar un estudio de las ventajas e inconvenientes de reescribir un programa de nuevo o de mantenerlo.

9.4.1.4.- Organización del equipo humano:

Es necesario organizar el equipo de mantenimiento, dividiendo claramente las actividades entre sus miembros y estableciendo los procedimientos de evaluación, control, supervisión e información de cada solicitud de mantenimiento.

Se pueden establecer las siguientes responsabilidades:

- Controlador del mantenimiento: persona que recibe la solicitud de mantenimiento y que asume la responsabilidad de su gestión y seguimiento integral.
- Supervisor del sistema software: persona encargada de conocer la aplicación a mantener y de informar sobre cada solicitud de mantenimiento que le afecte.
- Gestor de la configuración: persona que mantiene actualizada la configuración del software.
- Desarrollador de mantenimiento: persona que realiza los cambios en la aplicación.

9.4.1.5.- Documentación de los cambios:

Las solicitudes de mantenimiento deben hacerse mediante un formulario estandarizado. El equipo de mantenimiento debe elaborar un *informe de cambios* para cada solicitud.

Este informe de cambios debe incluir:

1.- Información del programa.
2.- Tamaño (LDC) del programa fuente.
3.- Tamaño del ejecutable.
4.- Lenguaje de programación utilizado.
5.- Fecha de instalación del programa.
6.- Número de ejecuciones del programa desde la instalación.
7.- Número de fallos.
8.- Número de sentencias añadidas, modificadas y eliminadas en el cambio.
9.- Número de personas-hora.
10.- Identificación de la persona responsable.
11.- Identificación de la solicitud de mantenimiento.
12.- Tipo de mantenimiento.
13.- Fechas de comienzo y final del mantenimiento.
14.- Beneficios netos que supone el cambio.

9.4.2.- SOLUCIONES TÉCNICAS:

Son de dos tipos: herramientas y métodos.

Las herramientas sirven para soportar de forma efectiva los métodos; han sido diseñadas para que el equipo de mantenimiento comprenda el programa y pruebe sus modificaciones asegurando que no han introducido errores. Estas herramientas son:

formateador, analizador estático, estructurador, documentador, depurador interactivo, generador de datos de prueba y comparador.

Los principales métodos utilizados en el mantenimiento son:

Reingeniería: consiste en el examen y modificación de un sistema para reconstruirlo de una nueva forma. Rehacer algo que otro ha realizado tratando de reutilizar.
Ingeniería Inversa: proceso de analizar un sistema para identificar sus componentes e interrelaciones, así como crear representaciones del sistema en un nivel de abstracción más elevado. Reinterpretar un programa para documentarlo.
Reestructuración del software: consiste en la modificación del software para hacerlo más inteligible y más fácil de cambiar. No cambia el nivel de abstracción.
Transformación de Programas: método formal que parte de un programa ya existente para obtener un programa equivalente por medio de transformaciones sucesivas.

9.5.- MANTENIBILIDAD:

También denominada *facilidad de mantenimiento del software*, se define como la medida cualitativa de la facilidad de comprender, corregir, adaptar y/o mejorar el software.

Los factores que influyen en la mantenibilidad son:

- Falta de cuidado en la fase de diseño, codificación o prueba.
- Pobre configuración del producto software.
- Adecuada cualificación del equipo de desarrolladores del software.
- Estructura del software fácil de comprender.
- Facilidad de uso del sistema.
- Empleo de lenguajes de programación y sistemas operativos estandarizados.
- Estructura estandarizada de la documentación.
- Documentación disponible de los casos de prueba.
- Incorporación en el sistema de facilidades de depuración.
- Disponibilidad del equipo hardware para realizar el mantenimiento.
- Disponibilidad de la persona o grupo que desarrolló originalmente el software.
- Planificación del mantenimiento.

Existen distintas clases de métricas de la mantenibilidad:

- *De esfuerzo*: indican el tiempo dedicado a las diversas tareas.
- *De complejidad*.
- *De estructura*.

9.5.1.- DISTINTOS ENFOQUES PARA LA FACILIDAD DE MANTENIMIENTO:

Una primera aproximación externa sería medir el proceso de mantenimiento; si es efectivo, entonces el producto es mantenible. En esta línea se pueden utilizar medidas de la cualificación del equipo de mantenimiento, de las herramientas disponibles y de la madurez del proceso.

Otra aproximación a la cual se recurre a menudo consiste en identificar atributos internos del producto y determinar cuáles son predictivos. Se basa en la métrica "tiempo medio de cambio" (TMC); que es el tiempo que transcurre desde que se recibe la solicitud de cambio hasta que el sistema cambiado es de nuevo operativo. Esta medida se obtiene de las siguientes variables: tiempo que se tarda en analizar la solicitud de cambio, tiempo de especificación y diseño del cambio, tiempo activo para implementar el cambio y tiempo en probarlo y distribuirlo.

Cuantos más atributos cualitativos se gestionen durante el desarrollo del software, mejor será su facilidad de mantenimiento (mantenibilidad). Desde un enfoque de calidad, estos atributos son: fiabilidad, modularidad, facilidad de comprensión, facilidad de prueba y facilidad de expansión.

9.5.2.- IMPACTO DE LA TECNOLOGÍA DE LA ORIENTACIÓN A OBJETOS EN LA FACILIDAD DE MANTENIMIENTO:

El factor clave de la OO en la mantenibilidad es la herencia. La herencia hace que las dependencias entre los objetos sean difíciles de encontrar y analizar. Con una herencia de 3 niveles se puede obtener aproximadamente una mejora en la mantenibilidad del 20%; no obstante, en cuanto aumentan los niveles aumenta la complejidad y el nivel de comprensión del programa, facilitando de este modo la reutilización pero, por el contrario, reduciendo considerablemente la mantenibilidad.

Existen diversas métricas de software OO, de las que cabe destacar en relación a la mantenibilidad:

- Métodos ponderados por clase (MPC) y complejidad ciclomática; miden la complejidad de la clase.
- Respuesta para una clase (RPC) y tamaño de un método; miden la facilidad de comprensión.
- Árbol de profundidad de herencia (APH), miden la facilidad de mantenimiento de una clase.
- Porcentaje de comentarios; evalúa la reutilización, mantenibilidad y facilidad de comprensión del código.

9.6.- EL MANTENIMIENTO FUTURO:

La tendencia actual se dirige hacia los sistemas basados en componentes reutilizables COTS (commercial off-the-shelf). Esto conlleva importantes cambios en el proceso de desarrollo y mantenimiento del software, lo que requiere nuevas cualificaciones del personal:

1) Arquitectos del sistema.

2) Especialistas en cada área tecnológica.
3) Especialistas en integración y test.

Ante esta disyuntiva, las organizaciones de software deberán decidir entre desarrollar, reutilizar o comprar (contratar) software. Factores o criterios que influirán en la toma de decisión son el coste, la disponibilidad, la experiencia del desarrollador/vendedor/contratante, la conformidad con los requisitos, la posibilidad de cambios y su futuro soporte.

El mantenimiento culmina el proyecto principal, a la vez que rebaja la calidad (en otras ingenierías no). La mantenibilidad se centra en el diseño del software pensando en el mantenimiento posterior.

7.7. ESTÁNDARES:

- Para los procesos del ciclo de vida del software: ISO 12207, IEEE 1074.
- Para la calidad del software y sus métricas: ISO 9126, IEEE 1061.
- Para el mantenimiento del software: IEEE 1219.

7.7.1.- ISO 12207:

International Standard for Information Technology – Software Life Cycle Processes. Publicado en 1995, se define el proceso de mantenimiento como una parte esencial del ciclo de vida del software.

Procesos principales:

- Adquisición: Necesidad de comprar un sistema.
- Suministro: Consigo el producto (propuesta), identifico lo que quiero y los recursos necesarios.
- Desarrollo: Técnicas de Ingeniería de Software.
- Explotación: Utilización por parte de los usuarios del sistema.
- Mantenimiento.

Procesos soporte:

- Documentación: Conjunto de soportes que registran las actividades de planificación, diseño, desarrollo, producción de todos los documentos necesarios para los distintos actores del proyecto: director, ingeniero y usuario. Supone editar, distribuir y mantener.
- Gestión de la Configuración: Forma de cómo va a funcionar lo realizado. Identificar la configuración, controlarla y el calendario de entrega.
- Aseguramiento de la calidad: Metodología o proceso por el cual se tiene una razonable seguridad de que se cumplen los requisitos especificados y que se sigue el plan establecido. Puede ser interno; asegurándome que fabrico un producto correcto, o externo; asegurándome que me venden lo que he pedido y como lo he pedido.
- Verificación: Estar seguros de que se cumplen todos y cada uno de los requisitos (diferente a asegurar la calidad).
- Validación: Comprobar que el producto sirve para el uso proyectado.

- Revisión conjunta (demo): Un proyecto es cosa de tres: el que lo define, el que lo desarrolla y el que lo usa.
- Auditoría: Control externo.

Procesos de organización:

- Proceso de gestión: Analizar la táctica y estrategia de la organización.
- Infraestructura: Asegurar que todos los procesos de producción funcionen.
- Mejora: Proceso por el cual toda organización aprende del trabajo realizado (experiencia).
- Formación: Cursos.

Proceso de adaptación:

Manera de instanciar los conocimientos en el entorno concreto.

7.7.2.- IEE 1074:

Developing Software Life Cycle Processes. Publicado en 1995 como resultado de la revisión de una norma de 1991, se detalla el conjunto de actividades que aparecen en el desarrollo y mantenimiento del software; dependiendo de los procesos sean de gestión de proyectos, predesarrollo, desarrollo, postdesarrollo o integrales.

7.7.3.- ISO/IEC ESTÁNDAR 9126:

Software Product Evaluation: Quality Characteristics and Guidelines for their Use. Publicado en 1991, se divide en dos estándares separados:

- El nuevo ISO/IEC 9126, llamado Software Quality Characteristics and Metrics, al que nos referiremos a continuación.
- ISO/IEC 14598, llamado Software Product Evaluation.

La mantenibilidad se define como la capacidad de un producto software para ser modificado. Se subdivide en 5 subcaracterísticas:

- **Analizabilidad:** Capacidad del producto software de diagnosticar sus deficiencias o causas de fallos, o de identificar las partes que deben ser modificadas.
- **Cambiabilidad:** Capacidad del producto software de permitir implementar una modificación previamente especificada.
- **Estabilidad:** Capacidad del producto software de minimizar los efectos inesperados de las modificaciones.
- **Facilidad de prueba:** Capacidad del producto software de permitir evaluar las partes modificadas.
- **Conformidad:** Capacidad del producto software de satisfacer los estándares o convenciones relativas a la mantenibilidad.