Compendio de Ingeniería del Software

Glosario

Rev.0.04 Junio 2006

.

Contenido

Introducción

Este glosario forma parte del proyecto CIS (Compendio de Ingeniería del Software).
En el se recopilan y definen términos propios del ámbito de la Ingeniería del Software.

Ámbito.

Términos notables de la Ingeniería del software.
No se trata de un diccionario informático, sino una guía de los términos, que por la frecuencia de su uso, o por su relevancia en la Ingeniería del software, deberían ser conocidos por los profesionales relacionados con las actividades de la Ingeniería del Software.

Propósito

Estandarización de los términos empleados en la práctica de la ingeniería del software.
Unificación de conceptos, significados y acepciones asociadas a los términos técnicos.

Sugerencias y contribuciones

Por favor envíeme sus sugerencias a : jpalacio<a>navegapolis.net
En el campo “asunto” del mensaje, además del asunto incluya la cadena: *CIS*

Glosario

Adaptabilidad. Facilidad con la que un sistema o un componente puede modificarse para corregir errores, mejorar su rendimiento u otros atributos, o adaptarse a cambios del entorno. Ver también: **escalabilidad**.

Análisis de requisitos. (1) Proceso de estudio de las necesidades del usuario para conseguir una definición de los requisitos del sistema o del software.
(2) Proceso de estudiar y desarrollar los requisitos del sistema o del software.

Aplicación de software. Software diseñado para satisfacer las necesidades de un usuario. Contrasta con: **software de soporte**; **software de sistema**.

Ciclo de vida. Periodo de tiempo que comienza con la concepción del producto de software y termina cuando el producto esta disponible para su uso.
Normalmente, el ciclo de vida del software incluye las fases de concepto, requisitos, diseño, implementación, prueba, instalación, verificación, validación, operación y mantenimiento, y, en ocasiones, retirada. Nota: Esta fases pueden superponerse o realizarse iterativamente.

CMM. Siglas de “Capability Maturity Model”, modelo desarrollado por SEI (Software Engineering Institute) en 1990, para la evaluación y mejora de los procesos.
El primer modelo desarrollado para evaluar y mejorar los procesos fue el SW-CMM, por lo que muchas veces se hace referencia a el coloquialmente como “CMM”.
En la actualidad los modelos de evaluación y mejora desarrollados y mantenidos por SEI son: P-CMM (People Capability Maturity Model), SA-CMM (Software Acquisition Capability Maturity Model).
Con la aparición en 2001 de los modelos *CMMI*, SEI ha dejado de mantener desde finales de 2004 los siguientes modelos CMM, por haberse integrado en los nuevos CMMI: SW-CMM (Capability Maturity Model for Software), SE-CMM (Systems Engineering Capability Maturity Model), IPD-CMM (Integrated Product Development Maturity Model).

CMMI Siglas de “Capability Maturity Model Integration”, modelos desarrollados por SEI que integran varias disciplinas: Desarrollo de software, Ingeniería de sistemas, Integración de productos y procesos de desarrollo.

COCOMO. (Construvtive Cost Model) Modelo constructivo de costes, desarrollado por B.W. Boehm a finales de los 70, y expuesto en su libro “Software Engineering Economics”. Es una jerarquía de modelos de estimación de costes que incluye los sub-modelos: básico, intermedio y detallado.

Codificación. (1) Proceso de descripción de un programa de ordenador en un lenguaje de programación.
(2) Transformación del diseño lógico y demás especificaciones de diseño en un lenguaje de programación.

Comité de gestión de configuración (CGC). Grupo de personas responsable de evaluar y aprobar cambios propuestos a elementos de configuración y garantizar la implementación de los cambios.

Compatibilidad. (1)Preparación de dos o más componentes o sistemas para llevar a cabo sus funciones mientras comparten el mismo entorno de hardware o software.
(2) Capacidad de dos o más sistemas o componentes para intercambiar información.

Complejidad ciclomática. Métrica que evalúa la complejidad del código. Los sistemas de software con puntos de excesiva complejidad ciclomática presentan un código con mayor dificultad de mantenimiento.

Componente. Una de las partes que forman un sistema. Un componente puede ser hardware, software, y puede a su vez subdividirse en otros componentes.

ConOps: v. **Descripción del sistema**.

CPM. (Critical Path Method) Método para el control y la optimización de los costes de operación mediante la planificación adecuada de las actividades que componen un proyecto. Fue desarrollado en 1957 en los Estados Unidos por un centro de investigación de operaciones para la firma Dupont y Remington Rand.
Actualmente se utilizan sus principios en combinación con los del método *PERT* en lo que se conoce como *PERT/CPM*

Crisis del software. Término acuñado en 1968, en la primera conferencia de la NATO sobre desarrollo de software, con el que se identificaron los problemas que surgían en el desarrollo de sistemas de software.

Crystal Methods. Metodología heterodoxa para desarrollo de software, creada por Alistair Cockburn, basada en su afirmación: “mucha gente piensa que el desarrollo de software es una actividad de ingeniería. Esa comparación es de hecho más perniciosa que útil, y nos lleva en una dirección equivocada.”

Descripción del sistema. Documento orientado al cliente que describe las características del sistema desde el punto de vista del usuario final. El documento se utiliza para coordinar conjuntamente los objetivos del sistema del usuario, cliente, desarrollador e intermediarios. También se denomina: **ConOps** (Conceptof Operation std. IEEE 1362). **Requisitos del sistema** (ISO IEC 12207 1995, 5.1.1.2)

Diseño. (1) Proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o de un componente.
(2) El resultado de este proceso.

Diseño de arquitectura. (1) Proceso que define una colección de componentes de software y hardware junto con sus interfaces, para definir el marco de desarrollo de un sistema. Ver también: **diseño funcional**.
(2) El resultado del proceso (1).

Diseño detallado.(1) Proceso de definición y ampliación del diseño preliminar de un sistema o de un componente hasta un grado de detalle suficiente para llevar a cabo la implementación.
(2) El resultado del proceso (1)

Diseño funcional. (1) Proceso de definición de las relaciones de trabajo entre los componentes de un sistema. Ver también: **diseño de arquitectura**.
(2) El resultado del proceso (1)

Diseño preliminar (1) Proceso de análisis de las alternativas de diseño y definición de la arquitectura, componentes, interfaces, estimación de tiempo y tamaño de un sistema o de un componente. Ver también: **Diseño detallado**.
(2) El resultado del proceso (1).

Disponibilidad. El grado con el que se mide la accesibilidad de un sistema o de un componente cuando es necesario su uso. Suele expresarse en términos de probabilidad. Ver también: **tolerancia a errores**, **tolerancia a fallos**, **robustez**.

DSDM v. Dynamic Systems Development Method

Dynamic Sistems Development Method (DSDM). Marco para desarrollo rápido de aplicaciones (RAD), muy popular en Gran Bretaña. Puede complementar a las metodologías XP, RUP o MSF. En esta metodología, al revés de lo que suele ocurrir en otras, el tiempo y los recursos se mantienen constantes, y se ajusta la funcionalidad.

Elemento de configuración. Parte de un desarrollo de software (planes, software, documentación de especificación y diseño, manuales, etc.) tratada como una unidad independiente en el proceso de gestión de configuración.

Escalabilidad. Facilidad con la que un sistema o un componente puede modificarse para aumentar su capacidad funcional o de almacenamiento. Ver también: **adaptabilidad**.

Especificación de interfaz. Documento que especifica las características de interfaz de un sistema o de un componente.

Especificación de requisitos de software. Documentación de requisitos fundamentales (necesarios, esenciales e indispensables) de funcionalidades, rendimiento, restricciones y atributos del software, y sus interfaces externas.
Su acrónimo inglés es SRS.

Estimación por analogía. Modelo de estimación de costes y recursos, basado en la comparación con proyectos de ámbitos y características similares, de los que se conocen sus costes reales por haberse terminado.

Evo. (Evolutionary Project Management) Metodología ágil creada por Tom Gilb. Es el método heterodoxo más veterano. También se le llama Evolutionary Delivery, Evolutionary Management, Requirements Driven Project Management y Competitive Engineering. Ofrece un planteamiento adaptativo orientado al cliente.

Extreme Programming. Metodología heterodoxa de programación. Es la más popular de las denominadas *metodologías ágiles*. Surgida a partir de la metodología de trabajo empleada Kent Beck, Wark Cunningham y Martin Fowler en el desarrollo del proyecto C3 para Chrysler. Extreme Programming (XP) se funda en cuatro valores: comunicación, simplicidad, feedback y coraje.

FDD. v. Feature Driven Development.

Feature Driven Development (FDD). *Metodología ágil* de desarrollo. No requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso. Se menciona por primera vez en el libro Java Modeling in Color with UML, de Meter Coad, Reic Lefebvre y Jeff DeLuca. Posteriormente DeLuca, Coad y Stephen Palmer lo desarrollaron más ampliamente. FDD consiste en cinco procesos secuenciales durante los que se diseña y construye el sistema: Desarrollo del modelo general – Construcción de la lista de rasgos – Planeamiento por rasgo – Diseño por rasgo – Construcción por rasgo.

Flexibilidad. Facilidad con la que un sistema o un componente puede modificarse para ser empleado con aplicaciones o en entornos distintos para los que fue construido.

Gestión de configuración. Disciplina que aplica la dirección y supervisión técnica y administrativa para: identificar y documentar las características funcionales y físicas de un elemento de configuración, controlar cambios, registrar cambios procesados, registrar el estado de la implementación, informar y verificar la conformidad con los requisitos especificados.

Gestión de procesos. Dirección, control y coordinación del trabajo realizado para desarrollar o producir un servicio.

Implementación. (1) Proceso de transformación de un diseño en componentes de hardware, software o de ambos. Ver también: **codificación**.
(2) El resultado del proceso (1).

Ingeniería del software. (1) Aplicación de procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de software.
(2) El estudio de la aplicación (1).

Interfaz. (1) Característica común en la información enviada.
(2) Componente de hardware o software que conecta dos o más componentes con el propósito de transmitir información entre ellos.
(3) Conexión de dos o más componentes con el propósito de transmitir información entre ellos.
(4) Empleado en la conexión (2)

Interfaz de usuario. Interfaz que permite la comunicación entre un usuario y un sistema, o los componentes de un sistema.

Línea de base. Conjunto de elementos de configuración, formalmente revisados y aprobados (para su uso interno o para entregar al cliente), que constituyen la base para el desarrollo posterior, y que sólo puede modificarse a través de procedimientos de cambio formales.

Mantenimiento. (1)Proceso de modificación de un sistema de software o de un componente, después de su puesta en funcionamiento para corregir fallos, mejorar el rendimiento u otros atributos, o adaptarlo a modificaciones del entorno. Ver también: **mantenimiento adaptativo**, **mantenimiento correctivo, mantenimiento perfectivo**.
(2) Proceso primario del modelo de ingeniería que desarrolla tareas de mantenimiento (1)

Mantenimiento adaptativo. Modificación de un sistema de software o de un componente, después de su puesta en funcionamiento, para adaptarlo a cambios del entorno. Contrasta con: **mantenimiento correctivo**; **mantenimiento perfectivo**.

Mantenimiento correctivo. Modificación de un sistema de software o de un componente, después de su puesta en funcionamiento para corregir fallos. Contrasta con: **mantenimiento adaptativo**; **mantenimiento perfectivo**.

Mantenimiento perfectivo. Modificación de un sistema de software o de un componente, después de su puesta en funcionamiento para mejorar el rendimiento u otros atributos. Contrasta con: **mantenimiento adaptativo**; **mantenimiento correctivo**.

Manual de diagnóstico. Documento con la información necesaria para ejecutar procedimientos de diagnóstico de un sistema o de un componente. Identifica errores de funcionamiento y establece cómo solucionarlos. Ver también: **manual de instalación**, **manual de operador**, **manual de programador**, **manual de soporte**, **manual de usuario**.

Manual de instalación. Documento que contiene la información necesaria para instalar un sistema o un componente, establecer los parámetros iniciales y preparar el sistema o componente para su uso. Ver también: **manual de diagnóstico**.

Manual de operador. Documento que contiene la información necesaria para iniciar y operar con un sistema o con un componente.
Nota: se establece diferencia entre un manual de operador y un manual de usuario, cuando en el sistema hay funciones propias de operación (cambio de discos o cintas, mantenimiento de base de datos, etc.) diferenciadas de las de uso normal del sistema para realizar las funciones que le son propias. Ver también: **manual de diagnóstico**, **manual de instalación**, **manual de programador**, **manual de usuario**.

Manual de programador. Documento que proporciona la información necesaria para desarrollar o modificar el software de un sistema. Ver también: **manual de diagnóstico**, **manual e instalación**, **manual de operador**, **manual de soporte**, **manual de usuario**.

Manual de soporte. Documento que contiene la información necesaria para mantener operativo un sistema durante su ciclo de vida. Ver también: **manual de diagnóstico**, **manual de instalación**, **manual de operador**, **manual de programador**, **manual de usuario**.

Manual de usuario. Documento que contiene la información necesaria para obtener de un sistema o de un componente los resultados deseados.
Nota: se establece diferencia entre un manual de operador y un manual de usuario, cuando en el sistema hay funciones propias de operación (cambio de discos o cintas, mantenimiento de

base de datos, etc.) diferenciadas de las de uso normal del sistema para realizar las funciones que le son propias. Ver también: **manual de operador**, **manual de instalación**.

Matriz de trazabilidad. Representación gráfica de las relaciones entre dos o más productos del proceso de desarrollo, generalmente identificadas en las intersecciones de lineas verticales y horizontales. Por ejemplo, para representar la relación entre los requisitos y el diseño de un componente del software.

Metodologías ágiles. Estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas; centradas en las personas o los equipos, iterativas, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa de cliente.

Microsoft Solutions Framework. (MSF) Marco para desarrollo de sistemas de software basado en principios, modelos, disciplinas, conceptos, prácticas y recomendaciones propias, derivadas de la experiencia de Microsoft. Se autodefine como “marco” y no como metodología, porque considera que no hay una única estructura de procesos válida para todos los proyectos. El marco MSF se adapta de forma flexible a las características de cada proyecto.

Con la aparición del producto Microsoft Visual Studio Team System, se ha actualizado MSF a la versión 4.0, produciendo dos variantes: MSF for Agile Software Development para el trabajo en entornos que emplean *metodologías ágiles*, y MSF for CMMI Process Improvement para el trabajo en entornos con el modelo *CMMI*.

Modelo de ciclo de vida. Representación del ciclo de vida del software.

Moore (Ley de). Gordon Moore, co-fundador de Intel afirmó en una entrevista a la revista Electronics, que el número de transistores por pulgada, implementados en los circuitos integrados se duplicaría cada año. Algo más tarde rectificó este plazo a 18 meses. Desde entonces hasta la fecha se viene cumpliendo esta progresión de crecimiento exponencial.

MSF. v. *Microsoft Solutions Framework*.

Nivel de integridad: Grado de daño que puede producir un fallo en un sistema. El estándar IEEE 1012-1998 define cuatro niveles de integridad para sistemas de software siendo el grado 1 el propio de sistemas cuyo fallo produce daños de escasa relevancia, y el 4 el que implica pérdidas de vida o graves pérdidas económicas o sociales.

Obtención. (aplicado a requisitos). Proceso en el que se implican las partes cliente y desarrolladora para descubrir, revisar, articular y comprender las necesidades y limitaciones que el sistema debe ofrecer a los usuarios.

OO. (Orientación por Objetos) Enfoque para el desarrollo de sistemas de software que representa el dominio de aplicación de forma natural y directa basándose en los objetos que se implican en dicho dominio.
Emplea diversos métodos para representar de forma abstracta los objetos, definiendo su estructura, comportamiento, agrupaciones, estados, etc.
Las estrategias de orientación por objetos han desarrollado metodologías tanto para requisitos, como para análisis, diseño y programación.
v. *OOA* (Análisis orientado por objetos), *OOD* (Diseño orientado por objetos), *OOP* (Programación orientada por objetos).

OOA (Object-Oriented Análisis) Análisis orientado por objetos. Método de análisis que examina los requisitos desde la perspectiva de clases y objetos encontrados en el vocabulario del dominio del problema. v. *OO*.

OOP (Object-Oriented Programming) Programación orientada por objetos. Método de implementación de los programas que los organiza como grupos cooperativos de objetos, cada uno de los cuales representa instancias de una clase, que a su vez forman parte de una jerarquía a través de relaciones de herencia. v. *OO*.

PERT. (Program Evaluation and Review Technique) Método para el control de los tiempos de ejecución de diversas actividades integrantes de proyectos. Fue desarrollado en 1957 por la armada de los Estados Unidos.
Actualmente se utilizan sus principios en combinación con los del método *CPM* en lo que se conoce como *PERT/CPM*

PERT/CPM. Método para el control de la ejecución de proyectos. Combina principios de los métodos PERT y CPM. Su desarrollo en un proyecto resulta útil para: conocer la probabilidad de cumplimiento de fechas, identificar las actividades con mayor potencial para retrasar el proyecto y evaluar las consecuencias de una desviación.

Plan de proyecto. Documento que describe el enfoque técnico y de gestión que seguirá un proyecto. Generalmente, el plan describe el trabajo a realizar, los recursos necesarios, los métodos a utilizar, los procesos a seguir, los programas a cumplir y la forma en la que se organiza el proyecto.

Proceso propio. Proceso definido en el modelo de ingeniería, y que junto con el resto de procesos del modelo constituye un valor activo de la organización (Know-how).

Producto de software. (1) Conjunto de programas, procedimiento y opcionalmente documentación asociada que se entrega al usuario como resultado.
(1) Uno de los elementos de (1).

Programa de ordenador. Combinación de instrucciones informáticas y definiciones de datos que permiten a un ordenador llevar a cabo tareas de control o de manipulación de información. Ver también **software**.

Programa principal. Componente de software, llamado desde un sistema operativo y que a su vez suele llamar a otros componentes de software.

Prototipado. Técnica de desarrollo consistente en la construcción de una versión preliminar de parte o de todo un sistema, para evaluar su viabilidad, funcionalidad, tiempos de respuesta, etc.

Prototipo. Versión preliminar de un sistema que sirve de modelo para fases posteriores.

Prueba de interfaz. Prueba cuya finalidad es evaluar el correcto intercambio de información y control entre componentes.

Prueba de sistema. Prueba cuya finalidad es evaluar el grado de conformidad con los requisitos de un sistema completo.

Prueba estructural. Prueba que centra su atención en la mecánica interna de un sistema o componente. Opuesto a: **prueba funcional**.

Prueba formal. Prueba ejecutada según planes y procedimientos de prueba revisados y aprobados por el cliente, usuario o personal de gestión. Opuesto a: **prueba informal**.

Prueba funcional. (1) Prueba que ignora la mecánica interna de un sistema o un componente y centra la atención sólo en las salidas generadas como respuesta a determinadas entradas y condiciones de ejecución.. Contrasta con: **prueba estructural**.
(2) Prueba cuyo fin es la evaluación del cumplimiento de un sistema o un componente con los requisitos funcionales.

Prueba informal. Prueba ejecutada según planes y procedimientos que no han sido revisados y aprobados por el cliente, usuario o personal de gestión. Opuesto a: **prueba formal**.

Puntos de función. Modelo de estimación basado en la perspectiva de la funcionalidad, sin contemplar detalles de la codificación. Se basa en una combinación de características del

sistema de software: entradas del usuario, salidas (presentadas) al usuario, consultas del usuario, archivos usados por el sistema e interfaces externos.

RAD. v. *Rapid Application Development*.

Rapid Application Development. (RAD) Denominación genérica para técnicas y herramientas de desarrollo de software que permiten el desarrollo rápido de aplicaciones.

Rational Unified Process (RUP). Proceso de Ingeniería del Software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades en las organizaciones de desarrollo de software. Se trata de un proceso integrado en un producto, desarrollado y mantenido por Racional Software, e integrado en su conjunto de herramientas de desarrollo. Se encuentra disponible a través de IBM.

Redundancia. Presencia de componentes auxiliares en un sistema para realizar funciones idénticas o similares a las de los componentes principales. Ver también: **redundancia activa**, **redundancia pasiva**.

Redundancia activa. Uso de elementos redundantes en operación simultánea para prevenir fallos. Contrasta con **redundancia pasiva**.

Redundancia pasiva. Uso de elementos redundantes que permanecen detenidos hasta que ocurre un fallo en el elemento principal. Contrasta con: **redundancia activa**.

Requisito. (1) Condición o facultad que necesita un usuario para resolver un problema.
(2) Condición o facultad que debe poseer un sistema o un componente de un sistema para satisfacer una especificación, estándar, condición de contrato u otra formalidad impuesta documentalmente.
(3) Documento que recoge (1) o (2).

Requisito de diseño. Requisito que especifica o impone condiciones al diseño de un sistema o de un componente. Contrasta con: **requisito funcional**, **requisito de implementación,** **requisito de interfaz**, **requisito de rendimiento**, **requisito físico**.

Requisito de implementación. Requisito que condiciona la codificación o la construcción de un sistema o de un componente. Contrasta con: **requisito de diseño**, **requisito funcional**, **requisito de interfaz**, **requisito de rendimiento**, **requisito físico**.

Requisito de interfaz. Requisito que especifica un elemento externo con el que un sistema o un componente debe interactuar; o que establece condiciones, formatos, tiempos u otros factores que deben respetarse en dicha interactuación. Contrasta con: **requisito de diseño**, **requisito funcional**, **requisito de implementación**, **requisito de rendimiento**, **requisito físico**.

Requisito de rendimiento. Requisito que impone condiciones sobre un requisito funcional. Por ejemplo los requisitos que especifican velocidad, precisión o uso de memoria. Contrasta con: **requisito de diseño**, **requisito de implementación**, **requisito de interfaz**, **requisito físico**, **requisito funcional**.

Requisito físico. Requisito que especifica las características físicas que debe presentar un sistema o un componente de un sistema; por ejemplo, material, longitud o peso. Contrasta con: **requisito de diseño**, **requisito de implementación**, **requisito de interfaz**, **requisito de rendimiento**, **requisito funcional**.

Requisito funcional. Requisito que especifica una funcionalidad que debe realizar un sistema o un componente. Contrasta con: **requisito de diseño**, **requisito de implementación**, **requisito de interfaz**, **requisito de rendimiento**, **requisito físico**.

Requisitos del sistema. V. **Descripción del sistema**.

Robustez. El grado de capacidad que presenta un sistema o un componente para funcionar correctamente frente a entradas de información erróneas, o carga de trabajo elevada. Ver también: **tolerancia a errores**; **tolerancia a fallos**.

RUP. v. Racional Unified Process

Scrum. *Metodología ágil*, aplicada originalmente por Jeff Sutherland y elaborada más formalmente por Ken Schwaber. Scrum aplica principios de control industrial, junto con experiencias metodológicas de Microsoft, Borland y Hewlet Packard.

SEI. (Software Engineering Institute) Fundación federal norteamericana para la investigación y desarrollo, cofinanciada por el Departamento de Defensa de los Estados Unidos y dependiente de la Universidad Carnegie Mellon.

Sistema. Conjunto de procesos, hardware, software, instalaciones y personas necesarios para realizar un trabajo o cumplir un objetivo.

Sistema de software. Conjunto de programas de ordenador, procedimientos y opcionalmente la documentación y datos asociados, necesarios para el funcionamiento de un sistema.

Sistema intensivo de software. Sistema en el que el principal componente es el software.

SLIM. (Software Lifecycle Management) Metodologías para estimaciones de duración, costes, control de proyectos y gestión de métricas. Desarrolladas por la comercial QSM

Software. Los programas de ordenador, procedimientos, y opcionalmente la documentación y los datos asociados que forman parte de un sistema.

Software de sistema. Software diseñado para facilitar o permitir la operación y el mantenimiento de un sistema informático; por ejemplo los sistemas operativos. Contrasta con **aplicación de software** y **software de soporte**.

Software de soporte. Software de ayuda para el desarrollo o mantenimiento de otro software; por ejemplo compiladores, editores y otras utilidades. **Contrasta con aplicación de software**; **software de sistema**.

SQA. (Software Quality Assurance) Se aplica a lo procesos o a las funciones encaminadas a garantizar que la organización realiza el trabajo de desarrollo, operación o mantenimiento de software conforme a los procedimientos y métodos establecidos para el proyecto.

Subsistema. *Sistema* subordinado a otro mayor.

SWEBOK. Siglas de: “Software Engineering Body Of Knowledge”, proyecto que tiene como finalidad definir y acotar las áreas de conocimiento que comprenden la Ingeniería del Software. En su desarrollo participan: IEEE, ISO/IEC/JTC1/SC, los principales autores de obras de Ingeniería del software: Steve Mc Connell, Roger Presuman e Ian Sommerville; así como importantes empresas: Racional, SAP, Boeing, Construx, MITRE, Raytheon.

TBD Siglas de la expresión inglesa “to be determined”. Generalmente aplicada a un requisito para indicar que está pendiente de determinar con el nivel de detalle requerido.

Trazabilidad. Grado de relación entre dos o más productos del proceso de desarrollo, especialmente productos que tienen una relación de predecesor – sucesor o de superior – subordinado con otro.

Trazabilidad de requisitos. Evidencia de una asociación entre un requisito y sus requisitos origen, su implementación y verificación.

Tolerancia a errores. Preparación de un sistema o de un componente para continuar su estado normal de operación, a pesar de la presencia de entradas erróneas. Ver también: **tolerancia a fallos**, **robustez**.

Tolerancia a fallos. Preparación de un sistema o de un componente para continuar su estado normal de operación, a pesar de la aparición de errores de hardware o de software. Ver también: **tolerancia a errores**, **robustez**.

Validación. Confirmación mediante examen y aportación de pruebas objetivas de que se cumplen los requisitos concretos para un uso determinado.

Verificación. Confirmación mediante examen y aportación de pruebas objetivas de que se cumplen los requisitos específicos.

Verificación y validación. Proceso que determina si los requisitos de un sistema o de un componente son completos y correctos, si los productos de cada fase cumplen los requisitos o condiciones marcados al inicio de la fase y si el sistema o componente final cumple con los requisitos especificados.

WBS. (Work Breakdown Structure). Método para representar jerárquicamente las partes de un proyecto, proceso o producto.

XP. v. *Extreme Programming*.

NOTAS