TECNOLÓGICO DE ESTUDIOS SUPERIORES DE CHALCO

HOJA DE EVIDENCIA DE EVALUACIÓN SUMATIVA

Carrera: Ingeniería en Sistemas Computacionales **Asignatura: Gestión de Proyectos**

Nombre del(a) alumno(a):

- Aguirre Velázquez Luis Raymundo
- Espinosa Sánchez Daniel Antonio
- Medina García José
- Olivares Vargas Luis Alberto
- Soriano López Alberto **Actividad correspondiente a: Unidad 2**

Nombre del profesor: Yahilt Hernández Hernández **Grupo:** 4701 **Fecha:**11/11/2020

Ada:
Lenguaje de programación que desarrolló el Departamento de Defensa estadounidense en la década de 1980 como un lenguaje estándar para desarrollar software militar. Se basa en investigación acerca de lenguajes de programación de la década de 1970 e incluye sentencias como tipos de datos abstractos y soporte para concurrencia. Todavía se utiliza en grandes sistemas militares y aeroespaciales complejos.

Administración de la configuración:
Proceso de administrar los cambios a un producto de software en evolución. La administración de la configuración implica planeación de la configuración, gestión de versiones, construcción de sistema y administración del cambio.

Administración del cambio:
Proceso para registrar, comprobar, analizar, estimar e implementar los cambios propuestos a un sistema de software.

Análisis estático:
Análisis basado en herramientas del código fuente de un programa para descubrir errores y anomalías. Las anomalías, como las asignaciones sucesivas a una variable sin uso intermedio, pueden ser indicadores de errores de programación.

Arquitectura cliente-servidor:
Modelo arquitectónico para sistemas distribuidos donde la funcionalidad del sistema se ofrece como un conjunto de servicios proporcionados por un servidor. A ellos acceden computadoras cliente que usan los servicios. Variantes de este enfoque, como las arquitecturas cliente-servidor de tres capas, usan múltiples servidores.

Arquitectura de referencia:
Arquitectura genérica idealizada que incluye todas las características que pueden incorporar los sistemas. Es una manera de informar a los diseñadores acerca de la estructura general de dicha clase de sistema en lugar de una base para crear una arquitectura de sistema específica.

Arquitectura de software:
Modelo de la estructura y organización fundamentales de un sistema de software.

Arquitectura dirigida por modelo (*model-driven architecture*, MDA):
Enfoque al desarrollo de software con base en la construcción de un conjunto de modelos de sistema, que pueden procesarse de forma automática o semiautomática para generar un sistema ejecutable.

Aseguramiento de la calidad (*quality assurance*, QA):
Proceso global para definir cómo puede lograrse la calidad del software y cómo la organización que desarrolla el software sabe que éste satisface el nivel de calidad requerido.

Ataque de negación de servicio:
Un ataque en un sistema de software basado en Web que trata de sobrecargar el sistema de forma que no pueda proporcionar su servicio normal a los usuarios.

BEA:
Un proveedor estadounidense de sistemas ERP.

Bomba de insulina:
Dispositivo médico controlado mediante software que puede suministrar dosis controladas de insulina a personas que sufren de diabetes. Se usó como estudio de caso en varios capítulos de este libro.

BPMN:
Business Process Modeling Notation (Notación para el Modelado de Procesos de Negocio). Una notación para definir flujos de trabajo.

C:
Lenguaje de programación que originalmente se desarrolló para implementar el sistema Unix. C es un lenguaje de implementación de sistema de nivel relativamente bajo, que permite el acceso al hardware del sistema y que puede compilarse a código eficiente. Se usa ampliamente para programación de sistemas de bajo nivel y desarrollo de sistemas embebidos.
C#:
Lenguaje de programación orientado a objetos, desarrollado por Microsoft, que tiene mucho en común con C++, pero que incluye características que permiten más comprobación de escritura a tiempo de compilación.

C++:
Lenguaje de programación orientado a objetos que es un súper conjunto de C.

CASE (*Computer-Aided Software Engineering*), ingeniería de software auxiliada por computadora:
El proceso de desarrollar software usando soporte automatizado.

Caso de confiabilidad:
Documento estructurado que se usa para respaldar las afirmaciones realizadas por un desarrollador de un sistema acerca de la confiabilidad de este último.

Caso de seguridad:
Argumento estructurado de que un sistema es seguro y/o está protegido. Muchos sistemas críticos deben tener casos de seguridad asociados que se valoran y aprueban mediante reguladores externos antes de que el sistema se certifique para su uso.

Caso de uso:
Especificación de un tipo de interacción con un sistema.

Ciclo de vida del software:
Con frecuencia se usa como otro nombre para el proceso de software; originalmente se acuñó para referirse al modelo en cascada del proceso de software.

Clase de objeto:
Una clase define los atributos y las operaciones de los objetos. Los objetos se crean en tiempo de ejecución al ejemplificar la definición de clase. El nombre de la clase se puede usar como un tipo de nombre en algunos lenguajes orientados a objetos.

CMM (Capability Maturity Model):
El modelo de madurez de capacidad del Software Engineering Institute, que se usa para valorar el nivel de madurez de desarrollo del software en una organización. Aunque lo desplazó el CMMI, todavía se usa ampliamente.

CMMI:
Enfoque integrado al modelado de madurez de capacidad de proceso con base en la adopción de buenas prácticas de ingeniería de software y gestión de calidad integrada. Apoya el modelado de madurez discreto y continuo, e integra modelos de madurez de sistemas y de procesos de ingeniería de software.

Cobertura de prueba:
Efectividad de las pruebas del sistema para probar el código de todo un sistema. Algunas compañías tienen estándares para cobertura de prueba (por ejemplo, las pruebas del sistema deben garantizar que todos los enunciados del programa se ejecuten al menos una vez).

Código de ética y práctica profesional:
Conjunto de lineamientos que establecen el comportamiento ético y profesional esperado por parte de los ingenieros de software. Lo definieron las principales sociedades profesionales estadounidenses (la ACM y el IEEE) y define el comportamiento ético bajo ocho encabezados: público, cliente y empleador, producto, juicio, administración, colegas, profesión y uno mismo.

COM+:
Modelo de componente y middleware de soporte diseñado para usar en plataformas de Microsoft; actualmente lo reemplaza .NET.

Componente:
Unidad de software independiente y portable que está completamente definido y al que se accede a través de un conjunto de interfaces.

Computación en nube:
La provisión de computación y/o servicios de aplicación a través de Internet con el uso de una “nube” de servidores de un proveedor externo. La “nube” se implementa usando un gran número de computadoras y tecnología de virtualización para usar de manera efectiva dichos sistemas.

Confiabilidad:
La confiabilidad de un sistema es una propiedad agregada que toma en cuenta la protección, fiabilidad, disponibilidad, seguridad y otros atributos del sistema. La confiabilidad de un sistema refleja el grado en el que los usuarios pueden confiar en él.

Construcción de sistema:
Proceso de compilar los componentes o las unidades que constituyen un sistema y los vinculan con otros componentes para crear un programa ejecutable. La construcción del sistema por lo general es automatizada, por lo que se minimiza la recopilación. Esta automatización puede instalarse en el sistema de procesamiento de lenguaje (como en Java) o puede implicar herramientas de software para soportar construcción del sistema.

CORBA (arquitectura común de intermediarios en petición de objetos, *Common Request Broker Architecture*):
Conjunto de estándares propuestos por el Object Management Group (OMG) que define modelos de objetos distribuidos y comunicaciones de objetos; es influyente en el desarrollo de sistemas distribuidos, pero actualmente se utiliza rara vez.

CVS:
Herramienta de software de fuente abierta ampliamente usada para gestión de versiones.

Desarrollo de software orientado a aspectos:
Un enfoque del desarrollo de software que combina desarrollo generativo y basado en componentes. Se identifican las competencias, intereses, asuntos o propiedades (*concerns*) transversales en un programa y la implementación de esas competencias se definen como aspectos. Los aspectos incluyen una definición de dónde se incorporan en un programa. Luego, un tejedor (*weaver*) de aspectos teje los aspectos en los lugares adecuados del programa.

Desarrollo dirigido por modelo (*model-driven development*, MDD):
Enfoque a la ingeniería de software centrado en modelos de sistema que se expresan en UML, en vez de utilizar código de lenguaje de programación. Esto extiende la MDA al considerar actividades distintas al desarrollo, como la ingeniería de requerimientos y las pruebas.

Desarrollo dirigido por pruebas:
Enfoque al desarrollo del software, donde se escriben pruebas ejecutables antes del código del programa. El conjunto de pruebas se corre automáticamente después de cada cambio al programa.

Desarrollo incremental:
Enfoque al desarrollo de software donde éste se entrega y despliega en incrementos.

Desarrollo iterativo:
Enfoque al desarrollo de software donde los procesos de especificación, diseño, programación y pruebas están entremezclados.

Desarrollo orientado a objetos:
Enfoque al desarrollo de software donde las abstracciones fundamentales en el sistema son objetos independientes. El mismo tipo de abstracción se usa durante la especificación, el diseño y el desarrollo.

Desarrollo rápido de aplicación (*rapid application development*, RAD):
Enfoque al desarrollo de software dirigido a la entrega rápida del software. Con frecuencia implica el uso de programación de bases de datos y herramientas de soporte de desarrollo, como generadores de pantalla y reportes.

Detección de fallas:
Uso de procesos y comprobación en tiempo de operación para detectar y eliminar fallas en el desarrollo de un programa antes de que den por resultado una falla en la operación del sistema.

Diagrama de clase:
Tipos de diagrama UML que muestran las clases de objetos en un sistema y sus relaciones.

Diagrama de estado:
Tipo de diagrama UML que muestra los estados de un sistema y los eventos que disparan una transición de un estado a otro.

Diagrama de secuencia:
Diagrama que muestra la secuencia de las interacciones requeridas para completar cierta operación. En UML, los diagramas de secuencia pueden asociarse con casos de uso.

Dinámica de evolución de programa:
Estudio de las formas en las que cambia un sistema de software en evolución. Se afirma que las leyes de Lehman gobiernan la dinámica de la evolución del programa.

Diseño de interfaz de usuario:
Proceso de diseñar la forma en la que los usuarios del sistema pueden ingresar a la funcionalidad de éste, y la forma en que se despliega la información producida por el sistema.

Disponibilidad:
La facilidad con la que un sistema proporciona servicios cuando se le solicitan. Por lo general, la disponibilidad se expresa como un número decimal, de manera que una disponibilidad de 0.999 significa que el sistema puede entregar servicios para 999 de 1,000 unidades de tiempo.

Dominio:
Área problemática o empresarial específica donde se usan los sistemas de software. Los ejemplos de dominio incluyen control en tiempo real, procesamiento de datos empresariales y conmutación de telecomunicaciones.

DSDM:
Método de desarrollo de sistema dinámico (Dynamic System Development Method); mencionado como uno de los primeros métodos de desarrollo ágiles.

EJB (Enterprise Java Beans):
Modelo de componentes basado en Java.

Entrega (*release*):
Versión de un sistema de software que se pone a disposición de los clientes del sistema.

Escenario:
Descripción de una forma típica en la que se usa un sistema o en la que un usuario realiza cierta actividad.

Etnografía:
Técnica de observación que puede usarse en la adquisición y el análisis de requerimientos. El etnógrafo se sumerge en el entorno del usuario y observa sus hábitos laborales cotidianos. A partir de las observaciones es posible inferir requerimientos para soporte de software.

Familia de aplicación:
Conjunto de programas de aplicación de software que tienen una arquitectura común y una funcionalidad genérica. Éstas se pueden ajustar a las necesidades de clientes específicos al modificar componentes y parámetros del programa.

Fiabilidad:
Capacidad de un sistema para entregar servicios de acuerdo con las especificaciones. La fiabilidad puede especificarse de manera cuantitativa como una probabilidad de falla a pedido o como la tasa de ocurrencia de fallas.

Flujo de trabajo:
Definición detallada de un proceso empresarial que tiene la intención de lograr cierta tarea. Por lo general, el flujo de trabajo se expresa gráficamente y muestra las actividades de proceso individual y la información que produce y consume cada actividad.

Framework de aplicación:
Conjunto de clases concretas y abstractas reutilizables que implementan características comunes a muchas aplicaciones en un dominio (por ejemplo, interfaces de usuario). Las clases en el Frameworks de aplicación se especializan e instancian para crear una aplicación.

Fuente abierta:
Enfoque al desarrollo de software donde el código fuente de un sistema se hace público y se alienta a usuarios externos a participar en el desarrollo del sistema.

Generador de programa:
Programa que genera otro programa a partir de una especificación abstracta de alto
nivel. El generador incrusta conocimiento que se reutiliza en cada actividad de generación.

Gestión de requerimientos:
Proceso de administrar los cambios a los requerimientos para asegurarse de que los cambios realizados se analizan adecuadamente y se rastrean a lo largo del sistema.

Gestión de versiones:
Proceso de gestionar los cambios a un sistema de software y sus componentes, de modo que sea posible conocer cuáles cambios se implementaron en cada versión del componente/sistema, y también para recuperar y recrear versiones anteriores del componente/sistema.

Gestión del riesgo:
Proceso de identificación de riesgos, valoración de su severidad, planeación de medidas para implementar en caso de que surjan riesgos, y monitorización del software y el proceso de software para detectar riesgos.

Gráfica de actividades (PERT):
Gráfica que usan los líderes de proyecto para mostrar las dependencias entre tareas que deben completarse. La gráfica muestra las tareas, el tiempo esperado para completarlas y sus dependencias mutuas. La ruta crítica es la ruta más larga (en términos del tiempo requerido para completar las tareas) a través de la gráfica de actividad. La ruta crítica define el tiempo mínimo requerido para completar el proyecto.

Gráfica de barras:
Gráfica que utilizan los líderes de proyecto para mostrar las tareas del proyecto, el calendario asociado con dichas tareas y las personas que trabajarán en ellas. Indica las fechas de inicio y fin de las tareas, y las asignaciones de personal, contra un cronograma.

Gráfica de Gantt:
Nombre alternativo para una gráfica de barras.

Herramienta CASE:
Una herramienta de software, como un editor de diseño o un depurador de programa, usada para apoyar una actividad en el proceso de desarrollo de software.

Ingeniería de sistemas:
Proceso que se ocupa de especificar un sistema, integrar sus componentes y probar
que el sistema satisface sus requerimientos. La ingeniería de sistemas se ocupa de todo el sistema sociotécnico (software, hardware y procesos operacionales), no sólo del software del sistema.

Ingeniería de software basada en componentes (CBSE, *Component-Based Software Engineering*):
Desarrollo de software mediante la composición de componentes de software independientes y portables que son congruentes con un modelo de componentes.

Ingeniería de software de cuarto limpio (Cleanroom):
Enfoque al desarrollo de software donde la meta es evitar introducir fallas en el desarrollo de software (por analogía con un cuarto limpio usado en la fabricación de semiconductores). El proceso implica especificación de software formal, transformación estructurada de una especificación a un programa, el desarrollo de argumentos correctos y pruebas estadísticas del programa.

Inspección de programa:
Revisión donde un grupo de inspectores examina un programa, línea por línea, con la intención de detectar errores. Con frecuencia las inspecciones se realizan con base en una lista de verificación de errores de programación comunes.

Interfaz:
Especificación de los atributos y las operaciones asociados con un componente de software. La interfaz se usa como medio para acceder a la funcionalidad del componente.

Interfaz de Programa de Aplicación (API):
Una interfaz, por lo general especificada como un conjunto de operaciones que permiten el acceso a la funcionalidad de un programa de aplicación. Esto significa que es posible que esta funcionalidad

sea llamada directamente por otros programas y no sólo accederse a ella a través de la interfaz de usuario.

ISO 9000/9001:
Conjunto de estándares o normas para procesos de gestión de la calidad definidos por la International Standards Organization (ISO). ISO 9001 es el estándar ISO que resulta más aplicable al desarrollo de software. Puede usarse para certificar los procesos de gestión de calidad en una organización.

Ítem de configuración:
Unidad legible por máquina, como un documento o un archivo de código fuente, que está sujeto a cambio y donde este último tiene que controlarse mediante un sistema de administración de la configuración.

J2EE:
Java 2 Platform Enterprise Edition. Complejo sistema middleware que apoya el desarrollo en Java de aplicaciones Web basadas en componentes. Incluye un modelo de componentes para componentes Java, APIs, servicios, etcétera.

Java:
Lenguaje de programación orientado a objetos usado ampliamente, diseñado por Sun con la intención de obtener independencia de plataforma.

Lenguaje de consulta estructurado (*Structured Query Language*, SQL):
Lenguaje estándar que se utiliza para programación de bases de datos relacionales.

Lenguaje de Modelado Unificado (*Unified Modeling Language*, UML):
Lenguaje gráfico que se utiliza en el desarrollo orientado a objetos e incluye varios tipos de modelos de sistema que ofrecen diferentes visiones de un sistema. El UML se convirtió en el estándar de facto para el modelado orientado a objetos.

Lenguaje de restricción de objetos (*Object Constraint Language*, OCL):
Lenguaje que es parte del UML, que se usa para definir predicados que se aplican a clases de objetos e interacciones en un modelo UML. El uso del OCL para especificar componentes es parte fundamental del desarrollo dirigido por modelo.

Leyes de Lehman:
Conjunto de hipótesis acerca de los factores que influyen en la evolución de sistemas de software complejos.

Línea de productos de software:
Véase familia de aplicación.

Make:
Una de las primeras herramientas de construcción de sistemas; todavía se usa ampliamente en sistemas Unix/Linux.

Manifiesto ágil:
Conjunto de principios que incluyen las ideas subyacentes de los métodos ágiles de desarrollo de software.

Mantenimiento:
Proceso de hacer cambios a un sistema después de ponerlo en operación.

Mejora de proceso:
Cambio en un proceso de desarrollo de software con la intención de hacer que dicho proceso sea más eficiente o mejore la calidad de sus resultados. Por ejemplo, si la intención es reducir el número de defectos en el software entregado, es posible mejorar un proceso al agregar nuevas actividades de validación.

Método estructurado:
Método de diseño de software que define los modelos de sistema que deben desarrollarse, las reglas y los lineamientos que deben aplicarse a dichos modelos, y un proceso a seguir en el desarrollo del diseño.

Métodos ágiles:
Métodos de desarrollo de software que se combinan para una entrega rápida del software. El software se desarrolla y se entrega en incrementos, y se minimizan la documentación del proceso y la burocracia. El foco del desarrollo está en al código en sí, y no en los documentos de apoyo.

Métodos formales:
Métodos de desarrollo de software donde el software se modela usando sentencias matemáticas formales como predicados y conjuntos. La transformación formal convierte este modelo en código. Se usa principalmente en la especificación y el desarrollo de sistemas críticos.

Métrica de control:
Métrica de software que permite a los administradores tomar decisiones de planeación con base en información acerca del proceso de software o el producto de software que se desarrollará. La mayoría de las métricas de control son métricas de proceso.

Métrica de predicción:
Métrica de software que se usa como base para realizar predicciones acerca de las características de un sistema de software, como su fiabilidad o mantenibilidad.

Métrica del software:
Atributo de un sistema o proceso de software que puede expresarse numéricamente y medirse. Las métricas de proceso son atributos del proceso, como el tiempo que tarda en completarse una tarea; las métricas de producto son atributos del software en sí, como el tamaño o la complejidad.

MHC-PMS:
Sistema de gestión de pacientes de atención a la salud mental; se usó como estudio de caso en varios capítulos.

Middleware:
Software de infraestructura en un sistema distribuido. Ayuda a gestionar las interacciones entre las entidades distribuidas en el sistema y las bases de datos del sistema. Ejemplos de middleware son un intermediario de solicitud de objetos y un sistema de gestión de transacciones.

Modelado algorítmico de costo:
Un enfoque a la estimación de costos del software donde se usa una fórmula para estimar el costo del proyecto. Los parámetros en la fórmula son atributos del proyecto y el software en sí.

Modelado de crecimiento de fiabilidad:
Desarrollo de un modelo de cómo cambia (mejora) la fiabilidad de un sistema conforme se prueba y se eliminan defectos del programa.

Modelo constructivo de costos (*Constructive Cost Modeling*, COCOMO):
Familia de modelos algorítmicos de estimación de costos. COCOMO se propuso por primera vez a principios de la década de 1980 y, desde entonces, se modificó y actualizó para reflejar la nueva tecnología y las cambiantes prácticas en la ingeniería de software.

Modelo de componentes:
Conjunto de estándares para implementación, documentación y despliegue de componentes. Cubre las interfaces específicas que pueden proporcionar un componente, nomenclatura, interoperación y composición de componentes. Los modelos de componentes brindan la base para que el middleware soporte componentes de ejecución.

Modelo de componentes CORBA:
Modelo de componentes diseñado para usar en la plataforma CORBA.

Modelo de dominio:
Definición de abstracciones de dominio, como políticas, procedimientos, objetos, relaciones y eventos. Sirve como base de conocimiento acerca de alguna área problema.

Modelo de madurez de proceso:
Modelo de la medida en la que un proceso incluye buenas prácticas y capacidades de medición que se integran para mejorar el proceso.

Modelo de madurez de capacidad del personal (*People Capability Maturity Model*, P-CMM):
Modelo de madurez de proceso que refleja cuán efectiva es una organización para administrar las habilidades, la capacitación y la experiencia de su personal.

Modelo de objeto:
Modelo de un sistema de software que se estructura y organiza como un conjunto de clases de objetos y las relaciones entre dichas clases. Pueden existir varias perspectivas diferentes del modelo, como una perspectiva de estado y una de secuencia.

Modelo de proceso:
Representación abstracta de un proceso. Los modelos de proceso pueden desarrollarse desde varias perspectivas y muestran las actividades implicadas en un proceso, los artefactos usados en éste, las restricciones que se aplican al proceso y los roles de las personas que lo ejecutan.

Modelo en cascada:
Modelo de proceso de software que comprende etapas de desarrollo discretas: especificación, diseño, implementación, pruebas y mantenimiento. En principio, una etapa debe completarse antes de que sea posible el avance a la siguiente etapa. En la práctica, existe significativa iteración entre etapas.

Modelo en espiral:
Modelo de un proceso de desarrollo donde el proceso se representa como una espiral; cada vuelta de la espiral incorpora las diferentes etapas del proceso. Conforme uno se mueve de una vuelta de la espiral a otra, se repiten todas las etapas del proceso.

.NET:
Marco de trabajo muy extenso que se usa para desarrollar aplicaciones para sistemas Microsoft Windows; incluye un modelo de componentes que define estándares para componentes en sistemas Windows y middleware asociado para apoyar la ejecución de componentes.

Object Management Group (OMG):
Grupo de compañías constituido con la finalidad de desarrollar estándares para el desarrollo orientado a objetos. Los ejemplos de estándares promovidos por el OMG son CORBA, UML y MDA.

Ocultamiento de información:
Uso de sentencias de lenguaje de programación para ocultar la representación de las estructuras de datos y controlar el acceso externo a dichas estructuras.

Patrón arquitectónico (estilo):
Descripción abstracta de una arquitectura de software que se ensayó y puso a prueba en algunos sistemas de software distintos. La descripción del patrón incluye
información acerca de dónde es adecuado usar el patrón y la organización de los componentes de la arquitectura.

Patrón de diseño:
Solución bien probada a un problema común que conjunta experiencia y buena práctica en una forma que pueda reutilizarse. Es una representación abstracta que puede ejemplificarse en varias formas.

Plan de calidad:
Plan que define los procesos y procedimientos de calidad que deben usarse. Esto implica seleccionar e instanciar los estándares para productos y procesos, y definir los atributos de calidad del sistema que son más importantes.

Prevención de fallas:
Desarrollo de software en tal forma que no se introduzcan fallas en el desarrollo de
dicho software.

Probabilidad de falla a pedido (*Probability Of Failure On Demand*, POFOD):
Métrica de fiabilidad que se basa en la probabilidad de que un sistema de software caiga cuando se hace una petición de sus servicios.

Proceso de software:
Conjunto de actividades y procesos relacionados implicados en el desarrollo y la evolución de un sistema de software.

Proceso Racional Unificado (*Rational Unified Process*, RUP):
Modelo de proceso de software genérico que presenta el desarrollo del software como una actividad iterativa de cuatro fases: concepción, elaboración, construcción y transición. La concepción establece un caso empresarial para el sistema, la elaboración define la arquitectura, la construcción implementa el sistema, y la transición implementa el sistema en el entorno del cliente.

Programación en pares:
Situación de desarrollo donde los programadores trabajan en pares, y no individualmente, para desarrollar código; es parte fundamental de la programación extrema.

Programación extrema (XP):
Método ágil de desarrollo de software usado ampliamente, que incluye prácticas como requerimientos basados en escenarios, desarrollo de primera prueba y programación en pares.

Propiedad emergente:
Propiedad que sólo se vuelve evidente una vez que se integran todos los componentes para crear el sistema.

Protección:
Capacidad de un sistema para operar sin falla catastrófica.

Pruebas de caja blanca:
Enfoque a las pruebas de programa, donde las pruebas se basan en el conocimiento de la estructura del programa y sus componentes. El acceso al código fuente es esencial para las pruebas de caja blanca.

Pruebas de caja negra:
Un enfoque a las pruebas donde los examinadores no tienen acceso al código fuente de un sistema o sus componentes. Las pruebas se derivan de la especificación del sistema.

Python:
Lenguaje de programación con tipos dinámicos, que es particularmente adecuado para el desarrollo de sistemas basados en Web; Google lo usa de manera extensa.

Reingeniería:
Modificación de un sistema de software para facilitar su comprensión y cambio. Con frecuencia, la reingeniería implica reestructuración y organización de software y datos, simplificación del programa y redocumentación.

Reingeniería, procesos empresariales:
Cambio de un proceso empresarial para satisfacer un nuevo objetivo organizacional, como costo reducido y ejecución más rápida.

Requerimiento funcional:
Enunciado de cierta función o característica que debe implementarse en un sistema.

Requerimiento no funcional:
Enunciado de una restricción o un comportamiento esperado que se aplica a un sistema. Esta restricción puede referirse a las propiedades emergentes del software que se desarrolla o al proceso de desarrollo.

Requerimientos de confiabilidad:
Requerimiento de sistema que se incluye para ayudar a lograr la confiabilidad requerida para un sistema. Los requerimientos no funcionales de confiabilidad especifican valores de atributo de confiabilidad; los requerimientos de confiabilidad funcional son requerimientos funcionales que especifican cómo evitar, detectar, tolerar o recuperarse de fallas en el desarrollo y la operación del sistema.

REST:
REST se deriva de Representational State Transfer (transferencia de estado representacional), que es un estilo de desarrollo basado simplemente en interacción cliente/servidor, y que usa el protocolo HTTP. REST se basa en la idea de un recurso identificable, que tiene una URI. Toda interacción con los recursos se basa en HTTP POST, GET, PUT y DELETE. Ahora se usa ampliamente para implementar servicios Web de carga baja.

Riesgo:
Resultado indeseable que plantea una amenaza al logro de cierto objetivo. Un riesgo de proceso amenaza la calendarización o el costo de un proceso; un riesgo de producto es un riesgo que puede significar que algunos de los requerimientos del sistema no se logren.

Ruby:
Lenguaje de programación con tipos dinámicos que es particularmente adecuado para programación de aplicaciones Web.

SAP:
Compañía alemana que desarrolló un sistema ERP bien conocido y ampliamente usado. También se refiere al nombre del sistema ERP en sí.

SCRUM:
Método de desarrollo ágil, que se basa en sprints: ciclos de desarrollo cortos. Scrum puede usarse como base para gestión de proyectos ágiles, junto con otros métodos ágiles como XP.

Seguridad:
Capacidad de un sistema para protegerse a sí mismo contra intrusión accidental o deliberada. La seguridad incluye confidencialidad, integridad y disponibilidad.

SEI:
Software Engineering Institute. Centro de investigación y transferencia tecnológica en ingeniería de software, fundado con la intención de mejorar el estándar de la ingeniería de software en las compañías estadounidenses.

Servicio Web:
Componente de software independiente al que puede accederse a través de Internet

usando protocolos estándar. Está completamente autocontenido sin dependencias externas. Se han desarrollado estándares basados en XML, como SOAP (Standard Object Access Protocol, protocolo estándar de acceso a objetos), para intercambio de información de servicio Web, y WSDL (Web Service Definition Language, lenguaje de definición de servicio Web), para la definición de interfaces de servicio Web. Sin embargo, el enfoque REST también puede usarse para implementar servicios Web.

Servidor:
Programa que proporciona servicio a otros programas (clientes).

Sistema crítico:
Sistema de cómputo cuya falla puede dar por resultado significativas pérdidas económicas, humanas o ambientales.

Sistema de planeación de recursos empresariales (*Enterprise Resource Planning*, ERP):
Un sistema de software a gran escala que incluye un rango de capacidades para soportar la operación de las empresas y que ofrece un medio para compartir información a través de dichas capacidades. Por ejemplo, un sistema ERP puede incluir soporte para proporcionar administración, fabricación y distribución en cadena. Los sistemas ERP se configuran con base en los requerimientos de cada compañía que usa el sistema.

Sistema de procesamiento de datos:
Sistema que se dirige a procesar grandes cantidades de datos estructurados. Dichos sistemas, por lo general, procesan los datos en lotes y siguen un modelo entrada-proceso-salida. Ejemplos de sistemas de procesamiento de datos son los sistemas de boletaje y facturación, y los sistemas de pago.

Sistema de procesamiento de lenguaje:
Sistema que traduce un lenguaje en otro. Por ejemplo, un compilador es un sistema de procesamiento de lenguaje que traduce el código fuente del programa en código objeto.

Sistema de procesamiento de transacciones:
Sistema que garantiza que las transacciones se procesen de tal forma que no puedan interferir entre sí y, por lo tanto, que la falla de transacción individual no afecte a otras transacciones o a los datos del sistema.

Sistema de tiempo real:
Sistema que debe reconocer y procesar eventos externos en “tiempo real”. La exactitud del sistema no sólo depende de lo que hace, sino también de qué tan rápido lo hace. Los sistemas de tiempo real por lo general se organizan como un conjunto de procesos secuenciales cooperativos.

Sistema distribuido:
Sistema de software donde los subsistemas o componentes de software se ejecutan en diferentes procesadores.

Sistema heredado:
Sistema socio técnico que es útil o esencial para una organización, pero que se desarrolló usando tecnología o métodos obsoletos. Puesto que los sistemas heredados con frecuencia realizan funciones empresariales críticas, deben mantenerse.

Sistema meteorológico a campo abierto:
Sistema para recopilar datos acerca de las condiciones meteorológicas en áreas remotas. Se usó como estudio de caso en varios capítulos de este libro.

Sistema par a par:
Sistema distribuido donde no hay distinción entre clientes y servidores. Las computadoras en el sistema pueden actuar como clientes y como servidores. Las aplicaciones pares a par incluyen compartición de archivos, mensajería instantánea y sistemas de apoyo a la cooperación.

Sistema socio técnico:
Sistema (incluyendo hardware y componentes de software) con procesos operacionales definidos, que siguen operadores humanos y que funciona dentro de una organización. Por lo tanto, recibe influencia de políticas, procedimientos y estructuras organizacionales.

sistemas basados en eventos:
Sistemas donde el control de la operación está determinado por eventos que se generan en el entorno del sistema. La mayoría de los sistemas de tiempo real son sistemas basados en eventos.

sistemas embebidos:
Sistema de software que se embebe en un dispositivo de hardware (por ejemplo, el sistema de software en un teléfono celular). Por lo general, los sistemas embebidos son sistemas de tiempo real y, por lo tanto, deben responder en forma oportuna a los eventos que ocurren en su entorno.

Subversión:
Herramienta de construcción de sistemas de fuente abierta, ampliamente utilizada, que está disponible para una variedad de plataformas.

Tasa de ocurrencia de fallas (*rate of ocurrence of failure*, ROCOF):
Métrica de fiabilidad que se basa en el número de fallas observadas de un sistema en un periodo de tiempo dado.

Tejedor de aspectos (*weaver*):
Un programa que por lo general es parte de un sistema de compilación que procesa un programa orientado a aspectos y modifica el código para incluir los aspectos definidos en los puntos especificados del programa.

Tiempo medio para falla (MTTF):
Tiempo promedio entre fallas de sistema observadas; se usa en especificación de fiabilidad.

Tipo de datos abstractos:
Un tipo que se define por sus operaciones y no por su representación. La representación es privada y sólo puede accederse a ella mediante las operaciones definidas.

Tolerancia a fallas:
Capacidad de un sistema para continuar en ejecución incluso después de que ocurran fallas.

Transacción:
Unidad de interacción con un sistema de cómputo. Las transacciones son independientes y atómicas (no se descomponen en unidades más pequeñas) y son una unidad fundamental de recuperación, consistencia y concurrencia.

Validación:
Proceso de comprobar que un sistema satisface las necesidades y expectativas del cliente.

Verificación:
Proceso de comprobación de que un sistema satisface sus especificaciones.

Verificación de modelo:
Método de verificación estático donde un modelo de estado de un sistema se analiza exhaustivamente con la intención de descubrir estados inalcanzables.

Workbench CASE:
Conjunto integrado de herramientas CASE que trabajan en conjunto para apoyar una actividad de proceso principal como el diseño de software o la administración de la configuración.

WSDL:
Notación basada en XML para definir la interfaz de servicios Web.

XML:
Extended Markup Language, es decir, lenguaje de marcas extensible. XML es un lenguaje de marca de texto que soporta el intercambio de datos estructurados. Cada campo de datos está delimitado por etiquetas que ofrecen información acerca de dicho campo. Ahora XML se usa ampliamente y se ha convertido en la base de protocolos para servicios Web.

XP:
Abreviatura utilizada comúnmente para Programación Extrema.

Z:
Lenguaje de especificación formal, basado en modelos, desarrollado en la Universidad de Oxford, en Inglaterra.