INSTITUTO TECNOLÓGICO SUPERIOR DE APATZINGÁN

INVESTIGACIÓN DOCUMENTAL

METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Alumno: Alejandra Virrueta Méndez

Carrera: Ingeniería en Informática.

Docente: Esmeralda Villegas Zamudio

Asignatura: Fundamentos de Investigación.

Apatzingan Michoacán Diciembre 2010

ÍNDICE._

PLANTEAMIENTO DEL PROBLEMA._ III

OBJETIVOS._ III

INTRODUCCIÓN._ IV

1._ LÍNEA DEL TIEMPO METODOLOGÍAS DE DESARROLLO DE SOFTWARE. 1

2. _MODELO EN CASCADA 3

3._ MODELO EN ESPIRAL 4

4._ RATIONAL UNIFIED PROCESS (RUP) 5

4.1 CARACTERÍSTICAS PRINCIPALES DEL PROCESO RUP._ 5

4.3._ FACES DE DESARROLLO DEL SOFTWARE 6

4.3.1._ Inicio 6

4.3.4._ Transición 7

5._ EXTREME PROGRAMMING (XP) 9

5.1._ BASES DE LA METODOLOGÍA XP. 9

6._ DESARROLLO MANEJADO POR RASGOS (FDD) 11

6.1._PROCESOS DE LA METODOLOGÍA FDD. 11

7._MICROSOFT SOLUTION FRAMEWORK (MSF) 13

7.1._ CARACTERÍSTICAS DE MSF 14

7.2._ MODELOS DE PLANIFICACIÓN EN MSF. 14

7.3._ FACES DE MSF._ 16

8._INCREMENTAL 18

9._RAPID APPLICATION DEVELOPMENT (RAD) 19

10._ OTROS ENFOQUES DE DESARROLLO DE SOFTWARE 21

11._ CONCLUSIONES._ 22

REFERENCIAS._ 23

Planteamiento del Problema._

En la actualidad se cuenta con diversos tipos de mitologías para el desarrollo de software, lo cual genera una gran problemática sobre cual debemos utilizar a la hora de diseñar un software, para esto es necesario conocer a fondo las diversas metodologías existentes, saber cómo funciona cada una, y así poder elegir correctamente la más adecuada según la necesidad que se tenga. En la presente investigación se dan a conocer las tecnologías de desarrollo de software más utilizadas así como su funcionamiento, con el fin de solucionar la problemática anteriormente mencionada.

Objetivos._

General.

Definir a fondo las Tecnologías de Desarrollo de Software, con el fin de ayudar a los diseñadores de software a la fácil elección de la metodología adecuada, según sea la necesidad que se tiene, es decir según lo que deba diseñar.

Específicos.

- Conocer a fondo cada Metodología de desarrollo de Software.
- Conocer los beneficios de cada mitología.
- Identificar la más apropiada para cada caso.

Introducción._

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo.

La metodología indica cómo hay que obtener los distintos productos parciales y finales.

Finalmente dependerá de la metodología utilizada los productos del proyecto, por esta razón es necesario, conoces a fondo cada una de ellas y poder diferenciar entre una y otra, para de este modo saber elegir la correcta en el momento de desarrollar un nuevo software, de otra manera el producto no será el mejor e incluso puede ser inútil.

1._ Línea del Tiempo Metodologías de Desarrollo de Software.

El desarrollo de los sistemas tradicionales de ciclo de vida se originó en la década de 1960 para desarrollar a gran escala funcional de sistemas de negocio en una época de grandes conglomerados empresariales. La idea principal era continuar el desarrollo de los sistemas de información en una muy deliberada, estructurada y metódica, reiterando cada una de las etapas del ciclo de vida. Los sistemas de información en torno a las actividades resueltas pesadas para el procesamiento de datos y rutinas de cálculo.

1970 s

- Programación estructurada desde 1969
- Programación estructurada Jackson desde 1975

1980 s

- Structured Systems Analysis and Design Methodology (SSADM) desde 1980
- Structured Analysis and Design Technique (SADT) desde 1980
- Ingeniería de la información (IE/IEM) desde 1981

1990 s

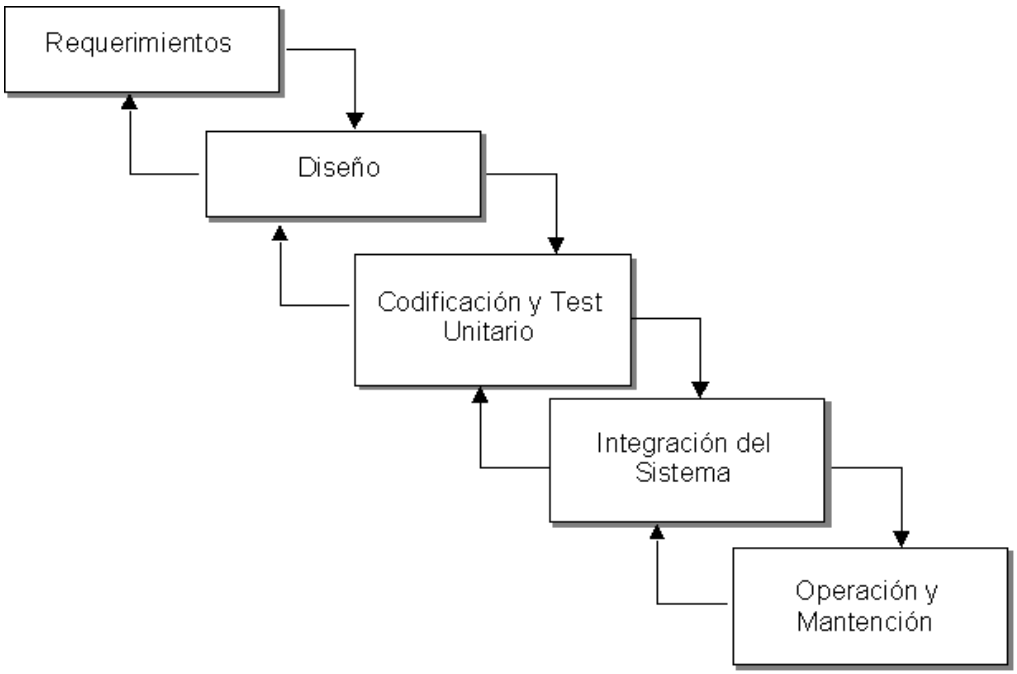
- Rapid application development (RAD) desde 1991.
- Programación orientada a objetos (OOP) a lo largo de la década de los 90's
- Virtual finite state machine (VFSM) desde 1990s
- Dynamic Systems Development Method desarrollado en UK desde 1995.
- Scrum (desarrollo), en la última parte de los 90's

Nuevo milenio

- Programación extrema desde 1999
- Enterprise Unified Process (EUP) extensiones RUP desde 2002
- Rational Unified Process (RUP) desde 2003.
- Constructionist design methodology (CDM) desde 2004 por Kristinn R. Thórisson
- Agile Unified Process (AUP) desde 2005 por Scott Ambler

2. _Modelo en Cascada

Este modelo utiliza tramos como puntos de transición y de carga. Al usar el modelo de cascada, se necesitaría completar un conjunto de tareas en forma de fase para después continuar con la fase próxima. El modelo en cascada trabaja perfectamente para los proyectos en los cuales los requisitos del proyecto se encuentran definidos claramente y no son obligados a futuras modificaciones. Ya que este modelo está compuesto por puntos de transición entre fases, se puede monitorear fácilmente ya que asigna responsabilidades definidas.



Esquema: 2 Estructura del modelo en cascada.

3._ Modelo en Espiral

Este modelo se basa en la necesidad continua de refinar los requerimientos para un determinado proyecto. El modelo espiral es eficaz cuando se utiliza para el rápido desarrollo de proyectos muy pequeños. Esta logra consigo el acercamiento entre el equipo de desarrollo y el cliente porque el cliente es implicado en todas las etapas proporcionando la regeneración de proyecto y la aprobación del mismo. De cualquier forma, el modelo en espiral no incorpora puntos de comprobación claros. Por lo tanto, el proceso de desarrollo puede llegar a ser caótico.

Esquema: 3 Modelo en espiral

4._ Rational Unified Process (RUP)

RUP es un proceso que define claramente quien, cómo, cuándo y qué debe hacerse; este aporta herramientas como los casos de uso, que definen los requerimientos además de permitir la ejecución iterativa del proyecto y del control de riesgos.

4.1 Características principales del proceso RUP.

Las características principales del proceso son:

- Guiado por los Casos de Uso
- Centrado en la Arquitectura
- Guiado por los Riesgos
- Iterativo

A través de un proyecto guiado por RUP, los requerimientos funcionales son expresados en la forma de Casos de Uso, que guían la realización de una arquitectura ejecutable de la aplicación. Además el proceso focaliza el esfuerzo del equipo en construir los elementos críticos estructuralmente y del comportamiento (llamados Elementos Arquitecturales) antes de construir elementos menos importantes. Finalmente RUP particiona el ciclo de vida en iteraciones que producen versiones increméntales de los ejecutables de la aplicación.

4.2 Practicas asociadas al proceso de Ingeniería de Software.

RUP implementa las siguientes mejores prácticas asociadas al proceso de Ingeniería de Software:

- Desarrollo Iterativo
- Manejo de los Requerimientos
- Uso de una Arquitectura basada en componentes
- Modelización Visual
- Verificación Continua de la Calidad
- Manejo de los Cambios

4.3. Faces de desarrollo del software.

La metodología RUP, llamada así por sus siglas en inglés *Rational Unified Process*, divide en 4 fases el desarrollo del software. Cada Fase tiene definido un conjunto de objetivos y un punto de control especifico:

4.3.1._ Inicio

a. Objetivos

- Definir el alcance del proyecto
- Entender que se va a construir

b. Puntos de Control

- Objetivo del proyecto

4.3.2._ Elaboración

a. Objetivos

- Construir una versión ejecutable de la arquitectura de la aplicación
- Entender cómo se va a construir

b. Puntos de Control

- Arquitectura de la Aplicación

4.3.3._ Construcción

a. Objetivos

- Completar el esqueleto de la Aplicación con la funcionalidad
- Construir una versión Beta

b. Puntos de Control

- Versión Operativa Inicial de la Aplicación

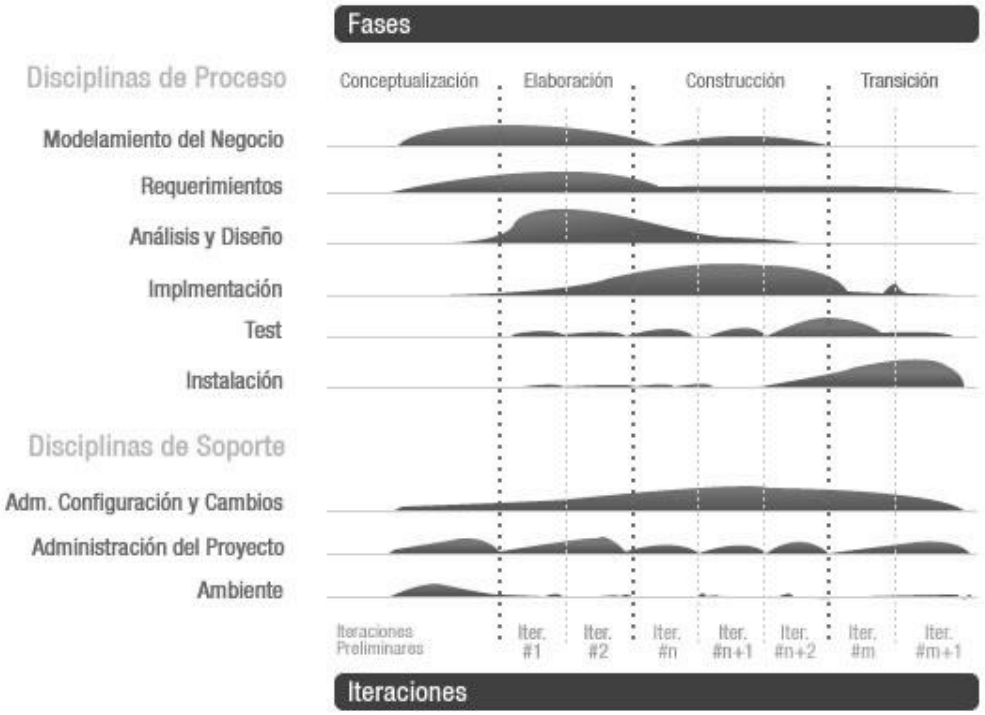
4.3.4._ Transición

a. Objetivos

- Hacer disponible la aplicación para los usuarios finales
- Construir la versión Final

b. Puntos de Control

- Liberación de la versión de la Aplicación



Esquema: 4 Fases e Iteraciones de la metodología RUP

5._ Extreme Programming (XP)

La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Esquema: 5 Metodología XP.

5.1. Bases de la Metodología XP.

Podemos encontrar que esta metodología está basada en:

- **Pruebas Unitarias**: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir.

- **Refabricación**: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares**: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Dentro de esta metodología existen las siguientes proposiciones:

- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso
- El costo del cambio no depende de la fase o etapa
- No introduce funcionalidades antes que sean necesarias
- El cliente o el usuario se convierten en miembro del equipo

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales

6._ Desarrollo Manejado por Rasgos (FDD)

El Desarrollo Manejado por Rasgos (FDD por sus siglas en inglés) fue desarrollado por Jeff De Luca y el viejo gurú de la OO Peter Coad. Como las otras metodologías adaptables, se enfoca en iteraciones cortas que entregan funcionalidad tangible. En el caso del FDD las iteraciones duran dos semanas.

6.1. Procesos de la metodología FDD.

El FDD tiene cinco procesos. Los primeros tres se hacen al principio del proyecto.

- Desarrollar un Modelo Global
- Construir una Lista de los Rasgos
- Planear por Rasgo
- Diseñar por Rasgo
- Construir por Rasgo

Los últimos dos se hacen en cada iteración. Cada proceso se divide en tareas y se da un criterio de comprobación.

Los desarrolladores entran en dos tipos: dueños de clases y programadores jefe. Los programadores jefe son los desarrolladores más experimentados. A ellos se les asignan rasgos a construir. Sin embargo ellos no los construyen solos. Solo identifican qué clases se involucran en la implantación de un rasgo y juntan a los dueños de dichas clases para que formen un equipo para desarrollar ese rasgo. El programador jefe actúa como el coordinador, diseñador líder y mentor mientras los dueños de clases hacen gran parte de la codificación del rasgo.

7._Microsoft Solution Framework (MSF)

Microsoft Solutions Framework (MSF) es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos, se ha convertido posteriormente en un modelo práctico que facilita el éxito de los proyectos tecnológico

Esquema: 6 Metodología MSF

7.1. Características de MSF.

MSF tiene las siguientes características:

- **Adaptable**: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable**: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible**: es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica**: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

7.2. Modelos de Planificación en MSF.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- **Modelo de Arquitectura del Proyecto**: Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

- **Modelo de Equipo**: Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

- **Modelo de Proceso**: Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

- **Modelo de Gestión del Riesgo**: Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

- **Modelo de Diseño del Proceso**: Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

- **Modelo de Aplicación**: Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

7.3. Faces de MSF.

Esta metodología se divide en diferentes fases las cuales son:

Fase 1: Visión. En esta fase el equipo y el cliente definen los requerimientos del negocio y los objetivos generales del proyecto. La fase culmina con el hito Visión y Alcance aprobados.

Fase 2: Planeación. Durante la fase de planeación el equipo crea un borrador del plan maestro del proyecto, además de un cronograma del proyecto y de la especificación funcional del proyecto. Esta fase culmina con el hito plan del proyecto aprobado.

Fase 3: Desarrollo. Esta fase involucra una serie de *releases* internos del producto, desarrollados por partes para medir su progreso y para asegurarse que todos sus módulos o partes están sincronizados y pueden integrarse. La fase culmina con el hito Alcance completo.

Fase 4: Estabilización. Esta fase se centra en probar el producto. El proceso de prueba hace énfasis en el uso y el funcionamiento del producto en las condiciones del ambiente real. La fase culmina con el hito *Release Readiness* aprobado.

Fase 5: Implantación: En esta fase el equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto, y obtiene la aprobación final del cliente. La fase termina con el hito Implantación completa.

Teniendo conocimiento sobre algunas de las metodologías existentes dentro del diseño de software y conociendo sus principales características, la metodología que se usa para el desarrollo del proyecto es MSF(*Microsoft Solutions Framework*).

8._Incremental

Provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto software reservando el resto de aspectos para el futuro.

Los principios básicos son:

- Una serie de mini-Cascadas se llevan a cabo, donde todas las fases de la cascada modelo de desarrollo se han completado para una pequeña parte de los sistemas, antes de proceder a la próxima incremental
- Se definen los requisitos antes de proceder con la evolutivo, se realiza un mini-Cascada de desarrollo de cada uno de los incrementos del sistema
- El concepto inicial de software, análisis de las necesidades, y el diseño de la arquitectura y colectiva básicas se definen utilizando el enfoque de cascada, seguida por iterativo de prototipos, que culmina en la instalación del prototipo final.

9._Rapid Application Development (RAD)

El *desarrollo rápido de aplicaciones* (RAD) es una metodología de desarrollo de software, que implica el desarrollo iterativo y la construcción de prototipos. El desarrollo rápido de aplicaciones es un término originalmente utilizado para describir un proceso de desarrollo de software introducido por James Martin en 1991.

Principios básicos:

- Objetivo clave es para un rápido desarrollo y entrega de una alta calidad en un sistema de relativamente bajo coste de inversión.
- Intenta reducir el riesgos inherente del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.
- Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de Interfaz gráfica de usuario (GUI), Computer Aided Software Engineering (CASE) las herramientas, los sistemas de gestión de bases de datos (DBMS), lenguajes de programación de cuarta generación, generadores de código, y técnicas orientada a objetos.

- Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.
- Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.
- En general incluye Joint application development (JAD), donde los usuarios están intensamente participando en el diseño del sistema, ya sea a través de la creación de consenso estructurado en talleres, o por vía electrónica.
- La participación activa de los usuarios es imprescindible.
- Iterativamente realiza la producción de software, en lugar de colgarse de un prototipo.
- Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

10._ Otros enfoques de desarrollo de software

- Metodologías de desarrollo Orientado a objetos, Diseño orientado a objetos (OOD) de Grady Booch, también conocido como Análisis y Diseño Orientado a Objetos (OOAD). El modelo incluye seis diagramas: de clase, objeto, estado de transición, la interacción, módulo, y el proceso.
- Top-down programming, evolucionado en la década de 1970 por el investigador de IBM Harlan Mills (y Niklaus Wirth) en Desarrollo Estructurado.

Proceso Unificado, es una metodología de desarrollo de software, basado en UML. Organiza el desarrollo de software en cuatro fases, cada una de ellas con la ejecución de una o más iteraciones de desarrollo de software: creación, elaboración, construcción, y las directrices. Hay una serie de herramientas y productos diseñados para facilitar la aplicación. Una de las versiones más populares es la de Rational Unified Process.

11._ Conclusiones._

Después de revisar los resultados de la presente investigación se obtuvieron las siguientes conclusiones:

- Las metodologías de desarrollo de Software se basan en diversas pruebas, y cada una tiene proceso divididos en fases.
- Cada metodología está diseñada para cumplir una necesidad especifica es decir, no todas tienen la misma funcionalidad, por ejemplo si el objetivo es la fácil y rápida creación de un programa sencillo se pude utilizar el modelo en espiral o el de cascada; pero si por el contrario se requiere el diseño de un programa tecnificado arquitectónico más complicado, lo ideal sería utilizar alguna metodología mas explicita como la RUP.
- Por otro lado cabe mencionar que es necesario conocer todas y cada una de estas metodologías de desarrollo, para poder ser acertados en la elección de la adecuada según nuestro objetivo.

Referencias._

http://www.getec.etsit.upm.es

Fuente: http://www.reynox.com

Fuente: http://www.microsoft.com

http://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software