Índice

INTRODUCCIÓN

A lo largo del desarrollo de un sistema software desde el estudio de requisitos hasta la implementación y puesta en marcha del mismo, se contemplan una multitud de tareas complejas. El software desde sus orígenes de desarrollo en los que se realizaban programas de forma poco sistemática, casi artesanal, hasta hoy en día en el que existen metodologías de desarrollo muy definidas, ha evolucionado en gran medida gracias a la ingeniería del software que pretende aplicar los principios de la ingeniería al desarrollo de software de forma que se sistematice la producción del mismo.

La crisis del software, término informático acuñado en 1968, en la primera conferencia organizada por la OTAN sobre desarrollo de software, se refiere a la dificultad en escribir programas libres de defectos, fácilmente comprensibles, y que sean
verificables. Las causas son, entre otras, la complejidad que supone la tarea de programar, y los cambios a los que se tiene que ver sometido un programa para ser continuamente adaptado a las necesidades de los usuarios. Esta conferencia supone un punto de inflexión en el desarrollo de software, ya que formalmente nace la ingeniería del software.

1. INGENIERÍA DEL SOFTWARE.

En los últimos años ha empezado a cambiar el enfoque que se le da al desarrollo de sistemas. Cada vez son más las organizaciones grandes y pequeñas que están adoptando un ciclo de vida uniforme y único para sus proyectos. Esto a veces se conoce como el plan del proyecto o metodología del desarrollo del sistema.

La crisis del software se refiere a la dificultad en escribir programas libres de defectos, fácilmente comprensibles, y que sean verificables. Las causas son, entre otras, la complejidad que supone la tarea de programar, y los cambios a los que se tiene que ver sometido un programa para ser continuamente adaptado a las necesidades de los usuarios.

La definición de ingeniería del software más aproximada es: aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software.

Su objetivo es facilitar el mantenimiento del software y aumentar la calidad de éste. La ingeniería del software por tanto estudiará el conjunto de procesos para el desarrollo de productos software según un enfoque ingenieril, considerando, además de la programación, otras actividades previas de análisis y diseño. Si se ignoran este tipo de actividades, en lugar de ingeniería del software se está realizando "artesanía del software".

2. CICLO DE DESARROLLO DEL "SOFTWARE".

2.1. Concepto de ciclo de vida.

El ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no cómo. Eso lo debería indicar la metodología.

El ciclo de vida abarca, por tanto, toda la vida del sistema, comenzando con su concepción y finalizando cuando ya no se utiliza.

Cada modelo de ciclo de vida va asociado a un paradigma de la ingeniería del software, es decir, a una serie de métodos, herramientas y procedimientos que debemos usar a lo largo de un proyecto.

2.2. Estándar de procesos del ciclo de vida software.

Las actividades que se pueden realizar durante el ciclo de vida del software se agrupan en procesos principales, procesos de soporte y procesos generales (de la organización), así como un proceso que permite adaptar el ciclo de vida a cada caso concreto.

Los procesos **principales** son aquellos que resultan útiles a las personas que inician o realizan el desarrollo, la explotación o el mantenimiento del software durante su ciclo de vida. Estas personas son los compradores, los suministradores, el personal de desarrollo, los operadores y el personal de mantenimiento del software.

Los procesos **organizacionales** los emplea una organización para llevar a cabo funciones tales como la gestión, la formación del personal o la mejora del proceso. Ayudan a establecer, implantar y mejorar la organización consiguiendo que sea más efectiva.

Los procesos de **soporte** sirven de apoyo al resto y se aplican en cualquier punto del ciclo de vida.

2.3. Relaciones de participantes en el ciclo de vida.

Es necesario comprender los procesos, las organizaciones y sus relaciones bajo diferentes puntos de vista como del contrato, de la gestión o dirección, de la explotación, de la ingeniería y del soporte.

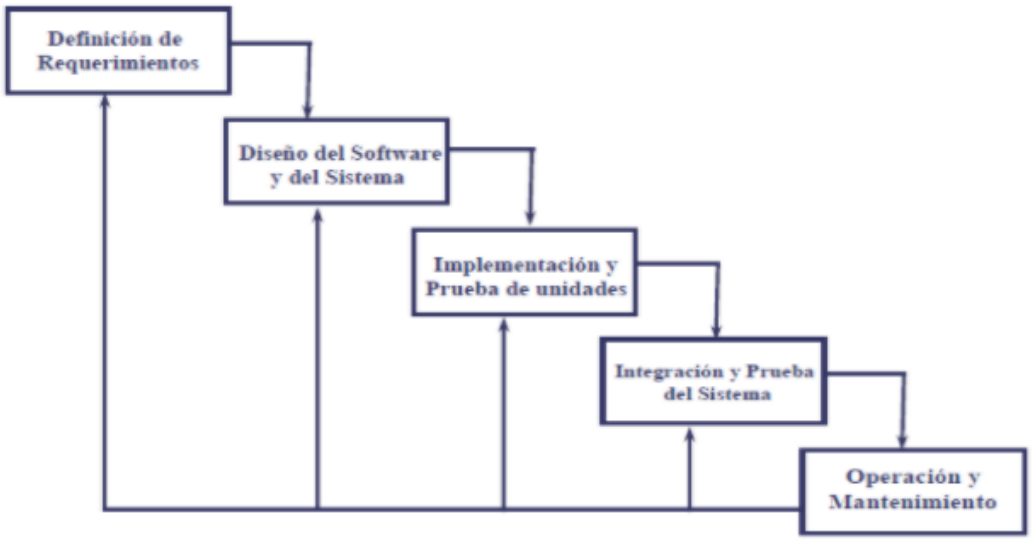
3. TIPOS DE CICLO DE DESARROLLO.

3.1. Modelo en cascada (waterfall).

El número de fases o etapas que se proponen en este ciclo suelen ser: análisis de requisitos del sistema, análisis de requisitos del software, diseño preliminar, diseño detallado, codificación, pruebas, explotación y mantenimiento.

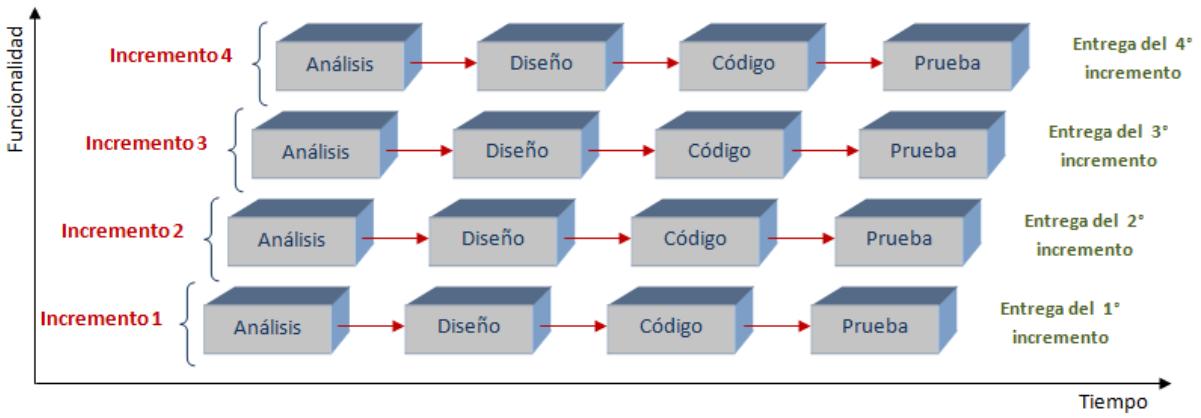
Algunas características de este ciclo son:

- Cada fase empieza cuando se ha terminado la fase anterior.
- Para pasar de una fase a otra es necesario conseguir todos los objetivos de la etapa previa.



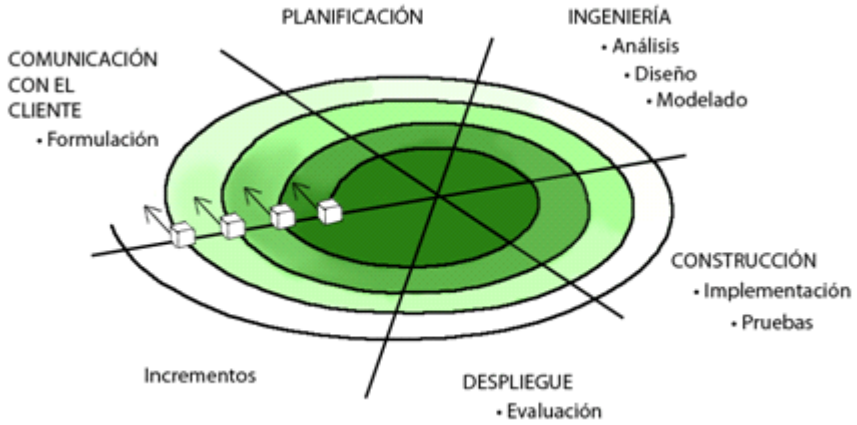
3.2. Modelo incremental.

Se va creando el sistema software añadiendo componentes funcionales al sistema (llamados incrementos). En cada paso sucesivo, se actualiza el sistema con nuevas funcionalidades o requisitos, es decir, cada versión o refinamiento parte de una versión previa y le añade nuevas funciones.



3.3. Modelo en espiral.

Consta de una serie de ciclos. Cada uno empieza identificando los objetivos, las alternativas y las restricciones del ciclo. Una vez evaluadas las alternativas respecto a los objetivos y teniendo en cuenta las restricciones, se lleva a cabo el ciclo correspondiente para, una vez finalizado, empezar a plantear el próximo.

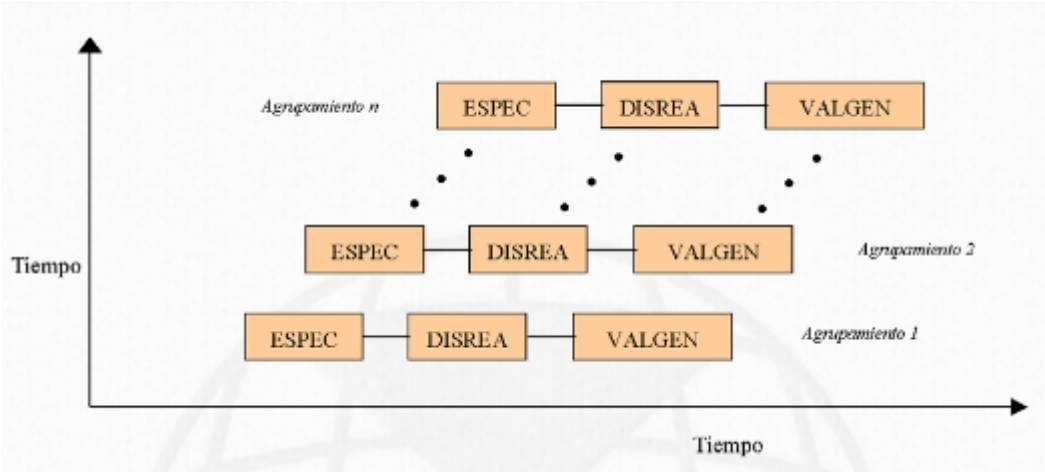


3.4. Modelos para desarrollo de sistemas orientados al objeto.

La tecnología de objetos se basa en "generalizar" los componentes para que sean reutilizables, lo que incrementa costes de desarrollo entre un 10 y un 50%, por lo que resulta imprescindible desarrollo que optimice esta inversión.

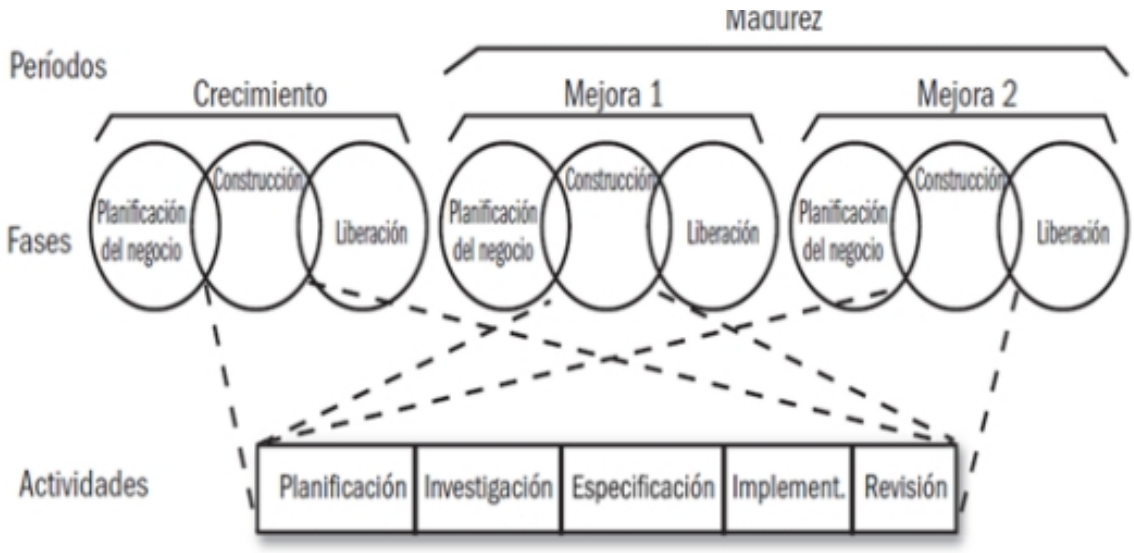
3.4.1. Modelo de agrupamiento (cluster)

El concepto clave de este modelo es el agrupamiento, que es un conjunto de clases relacionadas con un objetivo común (por ejemplo, un agrupamiento gráfico)



3.4.2. Modelo fuente.

Este modelo representa el alto grado de iteración y solapamiento que hace posible la tecnología de objetos.



3.4.3. Modelo remolino.

El modelo en cascada asume sólo una dimensión de iteración consistente en la fase del proceso, pero se pueden identificar otras dimensiones. Las diferentes dimensiones se pueden anidar de muchas maneras. Este proceso "fractal" (más que lineal), consistente en un desarrollo "multi cíclico", tiene la forma de un remolino en lugar de una cascada

3.4.4. Modelo pinball.

En este modelo la pelota representa un proyecto completo o un subproyecto y el jugador es el equipo de desarrollo.

Se procede de forma iterativa a encontrar clases, atributos, métodos e interrelaciones y definir colaboraciones, herencia, agregación y subsistemas. Por último se pasa a la programación, prueba e implementación. Realmente, como en el "pinball", los pasos se pueden tomar en cualquier orden y de forma simultánea.

4. METODOLOGÍAS DE DESARROLLO.

Atendiendo a una definición bastante genérica, podemos considerar una metodología de desarrollo como un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Representa el camino para desarrollar software de una manera sistemática.

4.1. Desarrollo convencional.

Este tipo de desarrollo no contemplaba la Surge la importancia del análisis y el diseño en desarrollo de un sistema. Hubo un cambio importante de papeles dentro de las organizaciones y se comenzó a hablar de analistas-programadores y de analistas de sistemas.

4.2. Desarrollo estructurado.

El nacimiento de las técnicas estructuradas se puede considerar un punto de partida en el que se pasa de la construcción de programas de una forma artesanal a una que sigue unos métodos de ingeniería sentando las bases para un desarrollo automatizado.

Surgen los siguientes conceptos:

- Programación estructurada. El enfoque de desarrollo estructurado comenzó con la programación. Se establecieron unas normas para la aplicación de las estructuras de datos y de control.
- Diseño estructurado. El enfoque estructurado se extiende a la fase de diseño. Se refina el concepto de modularidad, normalizando la estructura de un módulo de programa.
- Análisis estructurado(Top-down). Análisis dirigido a las especificaciones funcionales mediante gráficas, particionadas y mínimamente redundantes.

4.3. Desarrollo orientado al objeto.

El paradigma orientado a objetos, a diferencia del enfoque estructurado, trata los procesos y los datos de forma conjunta, es decir, modulariza tanto la información como el procesamiento. Al igual que los métodos estructurados (que se iniciaron con la programación), la orientación a objetos empieza con los lenguajes de programación orientados a objetos (LOO).

Mientras evolucionaban los lenguajes de alto nivel convencionales, surgidos de la tradición de FORTRAN (The IBM Mathematical **For**mula **Tran**slating) y ALGOL (Algorithmic Language), los investigadores trabajaban en una nueva clase de lenguajes de simulación y de creación de prototipos como el SIMULA y Smalltalk. En estos lenguajes se daba énfasis a la abstracción de datos y los problemas del mundo real se representaban como un conjunto de objetos de datos para los que se adjuntaba un conjunto de operaciones. Los primeros trabajos sentaron las bases de este enfoque al establecer la importancia sobre la abstracción, ocultación de la información y la modularidad, aspectos derivados del diseño estructurado.

La conferencia organizada por IBM "Object Oriented Programming Workshop" junto a la conferencia "First International Conference on Object-Oriented Programming Systems" en 1986 dieron lugar a que la orientación al objeto, alcanzara gran resonancia.

En el desarrollo al objeto se pueden distinguir las tres fases tradicionales: análisis, diseño e implementación. Sin embargo, hay que resaltar que estas fases se solapan y presentan unas fronteras más difuminadas que hay en el desarrollo tradicional.

Si atendemos al enfoque empleado en el proceso de desarrollo, podemos distinguir entre dos grandes corrientes:

- Metodologías dirigidas por los datos. Se basan principalmente en la parte estructural de los objetos, y son una extensión del modelo conceptual en el modelo E/R.
- Metodologías dirigidas por las responsabilidades. Representan el enfoque más purista de la orientación al objeto, y se centran en la responsabilidad del los objetos, esto es, las acciones que puede llevar a cabo un objeto.

Han habido una gran cantidad de notaciones diferentes pero se impone UML (Unified Modeling Language), un lenguaje gráfico de modelado para visualizar, especificar, construir y documentar las especificaciones de un sistema de software, y entre otros usos se emplea para el análisis y diseño de sistemas orientados a objetos. La metodología Métrica 3 tienen en cuenta la mayoría de las técnicas que contempla UML.

5. METODOLOGÍAS DE DESARROLLO UTILIZADAS EN LA UNIÓN EUROPEA.

5.1. Metodología MERISE.

Se compone de cuatro fases:

1. Estudio preliminar. Se analiza la situación existente para proponer una solución global atendiendo a los criterios de gestión, de la organización y decisiones adoptadas por el comité directivo del proyecto.
2. Estudio detallado. El objetivo de esta fase es definir, a nivel funcional, la solución que hay que implementar. Se obtendrán estudios detallados y los cuadernos de carga para la futura realización de los programas.
3. Implementación. El objetivo de esta etapa es realizar los programas que corresponden a las especificaciones detalladas; se divide en dos partes:
    a. Estudio técnico, donde se distribuyen los datos en ficheros físicos y los tratamientos en módulos de programas.
    b. Producción de programas, que permite codificar y verificar los programas mediante juegos de pruebas.
4. Realización y puesta en marcha. Se efectúa la implementación de los medios técnicos (instalación de los materiales, iniciación, captura de datos, etc.) y organizativos (formación del personal, lanzamiento de la aplicación), así como la recepción definitiva por parte del usuario.

5.2. Metodología SSADM.

Los aspectos claves de SSADM son:

1. Énfasis en los usuarios: sus requisitos y participación.
2. Definición del proceso de producción: qué hacer, cuándo y cómo.
3. Tres puntos de vista: datos, eventos, procesos.
4. Máxima flexibilidad en herramientas y técnicas de implementación.

5.3. Metodología MÉTRICA

El proyecto MÉTRICA, tiene como objetivo la creación de un marco metodológico común para la planificación y el desarrollo de sistemas de información de la Administración Pública española.

La última versión de esta metodología es la 3. Esta versión de MÉTRICA contempla el desarrollo de Sistemas de Información para las distintas tecnologías que actualmente están conviviendo y los aspectos de gestión que aseguran que un Proyecto cumple sus objetivos en términos de calidad, coste y plazos. La automatización de actividades propuestas en la estructura de MÉTRICA 3 es posible ya que sus técnicas están soportadas por una amplia variedad de herramientas de ayuda al desarrollo.

La metodología descompone cada uno de los procesos en actividades, y éstas a su vez en tareas. Para cada tarea se describe su contenido haciendo referencia a sus principales acciones, productos, técnicas, prácticas y participantes. El orden asignado a las actividades no debe interpretarse como secuencia en su realización, ya que éstas pueden realizarse en orden diferente a su numeración o bien en paralelo, sin embargo, no se dará por acabado un proceso hasta no haber finalizado todas las actividades del mismo determinadas al inicio del proyecto.

Así los procesos de la estructura principal de MÉTRICA Versión 3 son los siguientes:

PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN.
DESARROLLO DE SISTEMAS DE INFORMACIÓN.
MANTENIMIENTO DE SISTEMAS DE INFORMACIÓN.

En cuanto al Proceso de Desarrollo de Sistemas de Información, para facilitar la comprensión y dada su amplitud y complejidad se ha subdividido en cinco procesos:

ESTUDIO DE VIABILIDAD DEL SISTEMA (EVS).
ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI).
DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI).
CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI).
IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA (IAS).

La necesidad de acortar el ciclo de desarrollo de los sistemas de información ha orientado a muchas organizaciones a la elección de productos software del mercado cuya adaptación a sus requerimientos suponía un esfuerzo bastante inferior al de un desarrollo a medida, por no hablar de los costes de mantenimiento. Esta decisión, que es estratégica en muchas ocasiones para una organización, debe tomarse con las debidas precauciones, y es una realidad que está cambiando el escenario del desarrollo del software.

Otra consecuencia de lo anterior es la práctica, cada vez más habitual en las organizaciones, de la contratación de servicios externos en relación con los sistemas y tecnologías de la información y las comunicaciones, llevando a la necesidad de una buena gestión y control de dichos servicios externos y del riesgo implícito en todo ello, para que sus resultados supongan un beneficio para la organización.

MÉTRICA Versión 3 facilita la toma de decisión y la realización de todas las tareas que comprende el desarrollo de un sistema de información.

CONCLUSIÓN

Los cambios radicales en hardware a partir de la última mitad del siglo anterior causaron una forzada evolución del software, lo cual ha generado el establecimiento de modelos, estándares y redefinición de conceptos que ratifican un establecimiento cada vez más fuerte de la Ingeniería del Software a nivel mundial.

La gestión de proyectos de desarrollo de software es motor esencial para el éxito de cualquier proyecto de este tipo. La gestión debe fraccionarse en las etapas definidas claramente, manteniendo en cuenta los 4 requisitos indispensables: las personas, el producto, el proceso y el proyecto.

La programación orientada a objetos es una extensión actual de la tecnología que si bien ha evolucionado desde mediados del siglo pasado, presenta hoy día un enfoque nuevo y distinto al tradicional.

El diseño de la arquitectura es parte fundamental de los principios de la Ingeniería del Software y es único en el sentido de que se organiza en función de los objetos y clases que se definirán. De hecho, probablemente la parte más difícil del desarrollo de software orientado a objetos es la identificación de clases necesarias y la forma como interactúan entre sí.