UT02.
CICLO DE VIDA
METODOLOGÍAS DE DESARROLLO

ÍNDICE

Índice

1. INTRODUCCIÓN

En la presente Unidad de Trabajo se presentarán los porqués y el cómo del proceso de desarrollo software, las distintas fases que atraviesa todo el proceso desde la concepción hasta la retirada del producto realizado. Se introducen distintas teorías sobre las fases y metodologías de desarrollo tanto fijadas por estándares como otras de uso común.

La explicación de las metodologías pasa por la definición inicial de ciclo de vida, así como por la explicación de los distintos modelos de ciclo propuestos por organizaciones de estándares y autores de renombre.

Un **proceso software** es un conjunto de actividades y resultados que conducen a la creación de un producto software

1.1. El proceso de desarrollo de software en los Años 60 y 70

Desde una perspectiva histórica, el mercado de software apenas existía en los años 60-80 del siglo pasado, **el software básico se compraba al fabricante de la máquina** (Sistema Operativo, Compilador, Sort, etc), **había algún software de propósito general** de fabricantes independientes (Bases de Datos, algún programa de gestión de ficheros, y poco más), y **casi ningún Software de Aplicación.**

En una palabra, todos los programas de gestión de las empresas estaban hechos ex profeso **para las propias empresas**, y casi siempre **por personal de las propias empresas** que pertenecían al llamado entonces "Departamento de Proceso de Datos".

Las tareas eran casi artesanales, hechas a mano.

Generalmente, aunque no siempre, había una persona (**"*Analista Funcional o Jefe de Proyecto*"** era su denominación más habitual), que se ocupaba de la comunicación con usuarios, como Contabilidad, Cuentas Corrientes, Préstamos, etc. Esta figura, se encargaba de recibir peticiones del Usuario, proponer soluciones y dirigir al equipo de programadores que debían programarla. Una vez consensuada la Aplicación, realizaba el **Análisis Funcional** que era la descripción más o menos ordenada de las funciones que debe cubrir el Sistema de Información a diseñar.

Este proceso, como se comentaba antes, era artesanal o al menos bastante informal. No se seguía ningún método preestablecido ni se generaba una documentación según una estructura prediseñada al efecto. No había extensos documentos de justificación de gastos.

El personal encargado del desarrollo era personal de la propia empresa (generalmente bancos), de las áreas administrativas (cajeros, auxiliares, etc), que eran seleccionados para ser formados específicamente en el ordenador que había adquirido la empresa, durante varios meses.

El desarrollo se realizaba por **dos o tres programadores al servicio de cada jefe de proyecto** o analista funcional. No era difícil decidir quién se encargaba de qué ni hacer el seguimiento al proyecto.

El software se ponía en producción tras unas pruebas y **el mismo equipo daba el soporte (mantenimiento).**

Las bases de datos tenían escasas capacidades, se encontraban en los albores y la mayor parte de la información se almacenaba en ficheros secuenciales almacenados en cinta magnética.

Una de las "herramientas de diseño" era el diagrama de flujo u ordinograma, a veces mal llamado "organigrama" que definía las secuencias lógicas del programa. Se realizaba casi siempre a mano.

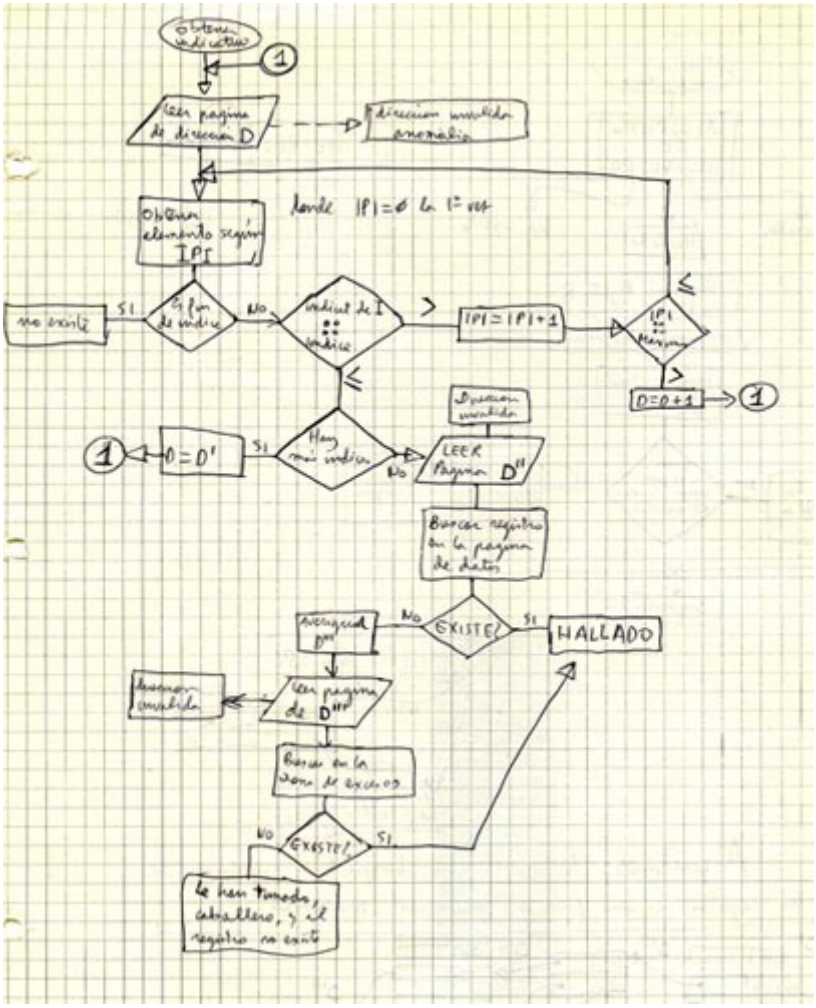


Ilustración 1: Diagrama de flujo

En cualquier caso, los saltos de un lado a otro del flujo, según las condiciones que se fueran satisfaciendo, se pintaban con flechitas, que tendían a entremezclarse y volar de arriba abajo, de derecha a izquierda y, en realidad, en todas direcciones … de ahí la denominación de "código spaghetti" para los programas resultantes (frente al código estructurado que surgirá posteriormente).

Una vez hecho el ordinograma (más o menos detallado, según la experiencia y habilidad del programador), se comenzaba a codificarlo, es decir, a traducirlo al lenguaje que el ordenador entiende (Algol, Cobol, Fortran o lo que fuera). El programa se escribía a mano en fichas que luego se transformaban en tarjetas perforadas. Esas tarjetas se traducían a código máquina durante la compilación, que era lo que se ejecutaba en las pruebas.

1.2. El proceso de desarrollo de software en los Años 80

1.2.1. Mejora tecnológica

A medida que la tecnología progresaba, fue posible desarrollar aplicaciones más complejas por los que se hizo posible que ciertas necesidades pudieran ahora por fin ser cubiertas mediante el desarrollo de software.

- ✓ Aparecen las **bases de datos relacionales**, que mejoran y racionalizan el tratamiento de datos, cambiando la filosofía en el diseño de aplicaciones.
- ✓ Aparece **la programación estructurada**.

El diseño de aplicaciones cada vez más complejas necesitaba **equipos de personas más numerosos** para su implementación.

En esa situación, **había que asegurarse**, en lo posible, de que tanto **el procedimiento de Diseño de las Aplicaciones** como **la Documentación asociada estuvieran formalizados**, uniformizados, reglamentados, constreñidos, de tal modo que cualquier persona pudiera leer, comprender y, llegado el caso, modificar cualquier Aplicación sin necesidad de apelar a la memoria o el conocimiento de su creador.

Lo realizado por un informático, cualquiera que sea su posición en la cadena de desarrollo (jefe de proyecto, analista, programador, etc, …), **debía quedar documentado mediante alguna manera que hiciera posible a otra persona retomar el trabajo del informático anterior**. Si al "*El Hombre-Aplicación*", esa persona que llevaba la Aplicación de Nóminas en la cabeza, igual que los "*Hombres-Libro*" le daba por irse de vacaciones, o enfermar, igual no se podía pagar la Nómina. Esta era una situación que se daba con alguna frecuencia, y que las empresas no estaban dispuestas a que volviera a ocurrir en el futuro.

Había que **implantar un procedimiento de Diseño, Construcción y Documentación** que permitiera eliminar a los "Hombres-Aplicación". Y para ello, se necesitaba, en primer lugar, un **Método** que dijera qué había que hacer, y qué no; en segundo lugar, unas **Técnicas** de uso común para realizar Análisis, Diseño, y Programación (aunque esta última ya estaba bastante formalizada); y en tercero, a ser posible, un **Sistema** de algún tipo que se asegure que las Normas del Método se cumplen y las Técnicas se aplican.

1.2.2. Nacimiento de las metodologías

Estas necesidades básicas son solucionadas, como habéis intuido, por las **Metodologías de Desarrollo de Software**, por las **Técnicas de Análisis y Diseño**, y por las **Herramientas CASE** (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), respectivamente.

Las metodologías, en su mayoría, **fueron promovidas por ciertos Gobiernos** para asegurar el cumplimiento de las especificaciones y, sobre todo, de los costes de los grandes proyectos contratados por la Administración.

Por otro lado, e**l mantenimiento de aplicaciones era** un serio problema, y tenía toda la pinta de convertirse en **El Problema**. Aquí tenemos un gráfico de los costes de cualquier gran Aplicación a lo largo del tiempo.

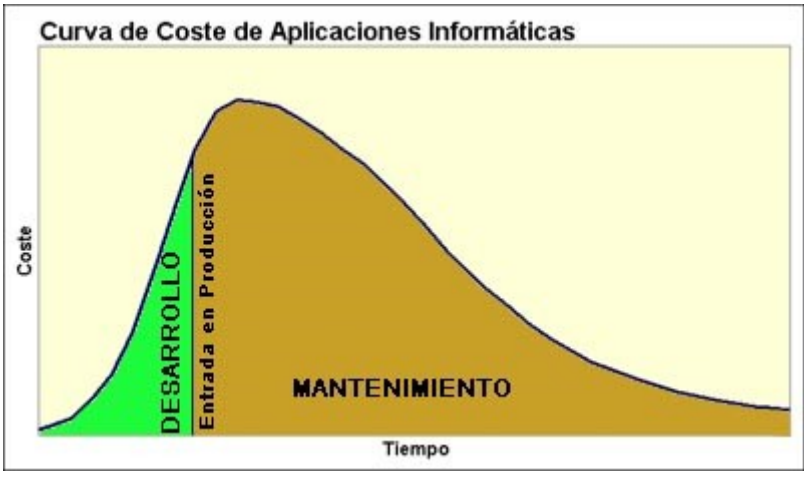


Ilustración 2: Curva de coste del Software

Una vez puesta en marcha (puesta en producción) la aplicación comienza su fase de mantenimiento (la zona naranja), que supondrá seguramente, a lo largo de toda la Vida de la

Aplicación, entre un 75% y un 90% del coste total dedicado a dicha Aplicación a lo largo de su vida útil.

La consecuencia a la que se llegó es que **no hay que escatimar esfuerzos durante la etapa inicial de Diseño y Construcción de la Aplicación para reducir todo lo posible el coste del Mantenimiento posterior.**

Una segunda vuelta de tuerca consistía en la búsqueda y **construcción de la herramienta CASE** ideal en la que, introducidas de alguna manera las premisas de la aplicación, se generase todo el diseño, documentación y el desarrollo, amén de facilitar el mantenimiento. Varias decenas de años después y unos cientos de herramientas, la búsqueda continúa aún hoy.

Desde luego ése es el ideal del empresario, encontrar una herramienta que pueda ser manejada por personal poco especializado (y barato), para generar complejas aplicaciones.

Las primeras herramientas fueron meras ayudas a la programación, ayudando a hacer el trabajo más rápido y facilitando la detección y eliminación de errores del código (depuración). Editores y depuradores de programas sobre pantalla se encargaban de esa labor.

→ Métodos o Técnicas y herramientas de diseño

Varios ingenieros de la computación estaban atacando ya (de hecho, llevaban algunos años en ello) el problema siguiente al de cómo realizar eficientemente la Programación, es decir: ¿**cómo realizar el Diseño de los Sistemas**? Hacer tanto el Diseño Técnico (Bases de Datos, Transacciones, Procesos Batch…) como el propio Análisis Funcional, de alguna manera ciertamente más formal que simplemente escribir la documentación

Ilustración 3: Peter Chen

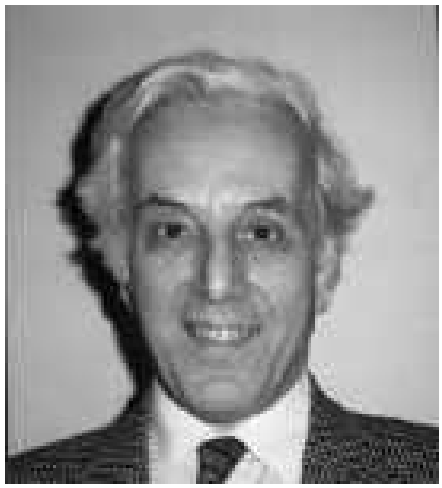


Ilustración 4: M. A. Jackson

Uno de los esfuerzos vino de Michael Jackson[1], que en 1983 publicó su libro "System Development", en el que presenta su método JSD de programación estructurada, aunque ya estaba dando cursos y seminarios sobre el asunto desde hacía un par de años.

En realidad JSD es una extensión de su método de Estructuración de Programas, esta vez orientado al Diseño de los Sistemas. No tuvo excesivo éxito (veremos que llegó algo tarde), pero nuevamente Jackson hizo una formalización extraordinaria: su método fue el primero que estableció que **los Sistemas deben ser diseñados a partir de las Salidas** (es decir, los resultados esperados). Todos los procesos, los datos necesarios, etc, son la consecuencia lógica de conocer para qué debe servir nuestro sistema, es decir, qué es lo que esperamos que haga el

[1] Evidentemente, no era familia del cantante

sistema, lo que debe de producir, en definitiva: las salidas. No tuvo el éxito debido a que llegó tarde.

Otra técnica de programación estructurada que partía de los datos fue **Warnierr-Orr.** Tanto JSD como Warnier reducían las estructuras o bloques programables a tres: **Iteración, Selección y Secuencia**, eliminando los saltos incondicionales, rupturas de la secuencia del programa a capricho o sentencias GOTO.

Anteriormente, Peter P. Chen había atacado con éxito la problemática de la modelización de los datos, mediante su técnica "Modelo de Entidad-Relación" (ERM), publicado inicialmente tan pronto como en 1976, pero que no tuvo aceptación real hasta que las Bases de Datos Relacionales no comenzaron a invadir las instalaciones de todo el mundo, a mediados de los ochenta. Esta forma de modelizar los datos fue adoptada por virtualmente todos los métodos de Desarrollo de Aplicaciones de entonces (y de ahora), debido a su consistencia y, sobre todo, a su facilidad de aprendizaje y comprensión.

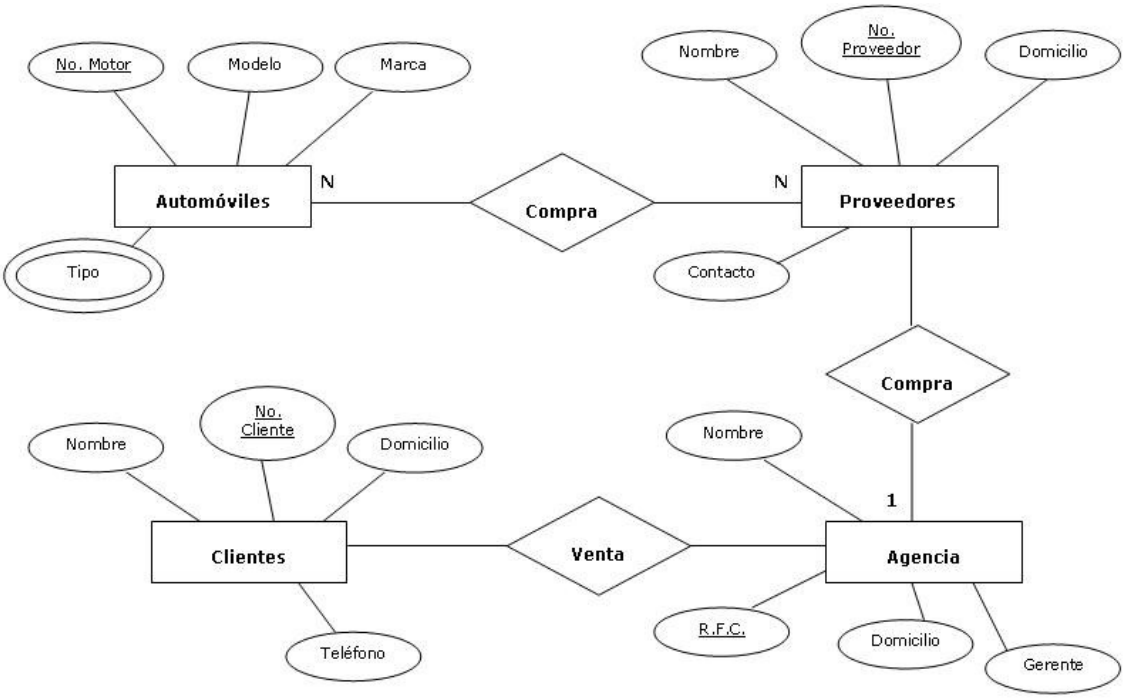


Ilustración 5: Modelo E-R extendido

Pero había otros gurús que pensaban justo lo contrario que Jackson y Chen: **que lo importante de las aplicaciones no son los datos, sino los Procesos**, es decir, conocido lo que hay que hacer, los datos necesarios para hacerlo se hacen evidentes, pues son el subproducto de los procesos.

Edward Yourdon y Tom de Marco propusieron esta Técnica de Análisis y Diseño Estructurado, que se basa en una aproximación top-down al problema del Diseño de los Sistemas de Información: comenzar con el diagrama más general, e ir descomponiendo el Sistema desde lo más general a lo más particular. El resultado son los Diagramas de Flujo de Datos (DFD's)

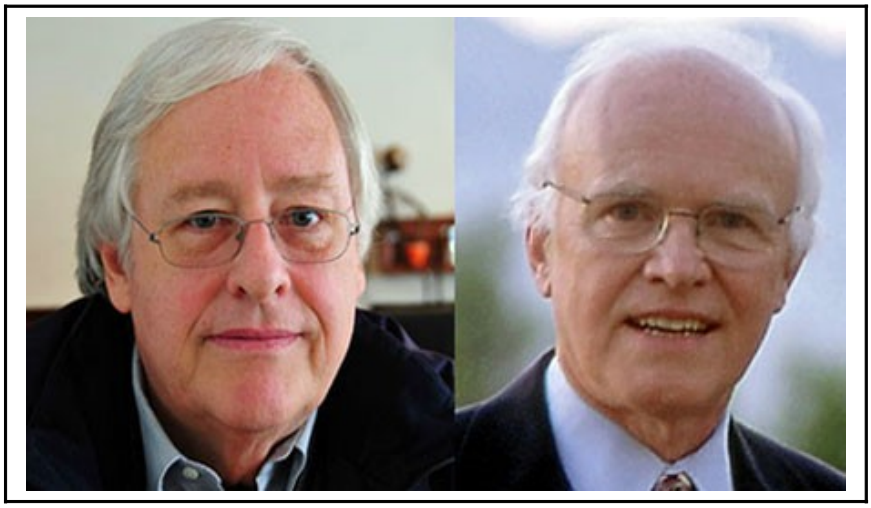


Ilustración 6: Ed Yourdon y Tom de Marco

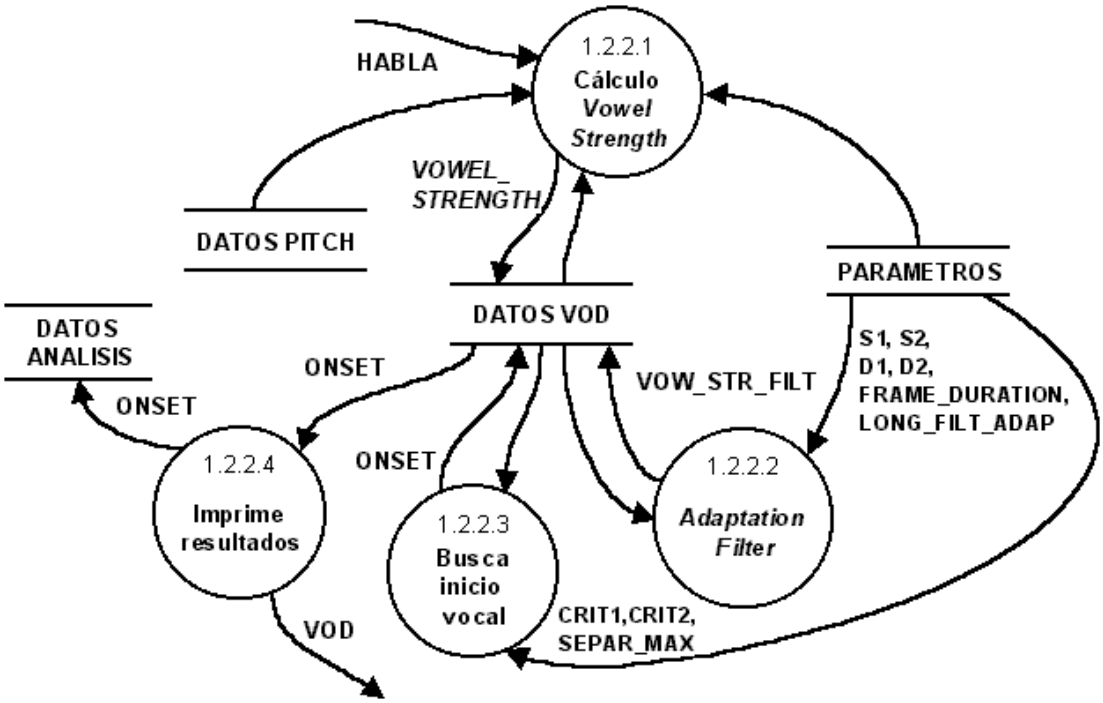


Ilustración 7: Ejemplo de un DFD

En los DFD, las burbujas indican una acción (véase que contienen siempre un verbo), esto es, una acción o mejor, **un proceso**. Las flechas son los flujos o movimientos de los datos, que pueden recibirse o enviarse a otros procesos o bien recuperarse o almacenarse en los "almacenes" indicados con dos líneas paralelas (en la figura anterior, PARAMETROS es un almacén de datos).

Uno de los personajes más importantes de la informática del siglo pasado, el británico James Martin, se adhirió pronto y de modo estusiasta a las técnicas de modelización, y creó, o quizá sólo la adoptó y dio a conocer, en 1981, la Metodología **IE** (Information Engineering), que tuvo casi de inmediato una muy buena aceptación. Usó las dos técnicas de modelización antes mencionadas: la de procesos, vía los DFD's, y la de datos, vía los ERD's.

Ilustración 8: James Martin

A raíz de la publicación de los métodos gráficos anteriores surgieron las herramientas CASE que facilitaban su uso de modo automatizado.

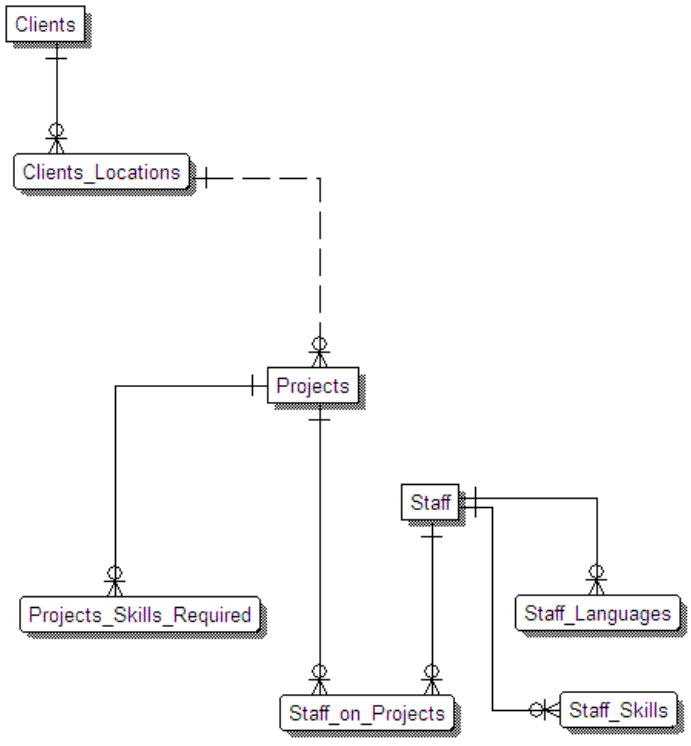


Ilustración 9: Notación Martin de un modelo de datos

→ Metodologías

Una metodología **abarcará todo el proceso completo** desde la concepción de la aplicación a desarrollar hasta su retirada, incluyendo la fase de mantenimiento, esto es, abarca todo el ciclo de vida del software **y especifica al detalle cada paso**.

A partir de aquí, en la meca de la informática, EEUU, se comienzan a utilizar metodologías de desarrollo, primero de forma tímida, luego con cada vez mayor aceptación.

En Europa, se desarrollaron mientras tanto unas metodologías globales que recogieron el paso a paso, usando diversas técnicas y métodos, del desarrollo y mantenimiento de un producto software. Nacieron de mano de las Administraciones Públicas de cada país.

En Francia existía desde hacía unos años (mediados de los setenta) la metodología Merise, en principio una relación (exhaustiva, eso sí) de pasos y tareas con alguna técnica manual sencilla, y que también a principios de los ochenta se enriqueció con las nuevas técnicas de modelización de Datos y Procesos; se trata de un método francés, pensado por y para franceses, y que tuvo pocos seguidores fuera del mundo francófono, pero como la Administración francesa exigió el uso de Merise para la contratación de proyectos informáticos desde muy pronto (principios de los ochenta), se convirtió en el standard *de facto* en el mercado francés en el final de la década de los ochenta.

En el Reino Unido, en la misma época, se estaba gestando SSADM, que adoptó entre sus técnicas a una mezcla de varios de los incipientes métodos de análisis estructurado del momento, en particular los de Yourdon, De Marco, Jackson y Constantine, que fueron incorporados desde el principio al método, se supone que tomando lo mejor de cada uno de ellos. A partir de 1983, también se hizo obligatorio el uso de SSADM para poder contratar proyectos para la Administración británica, por lo que fue ampliamente seguida en el Reino Unido, y después también en el resto de Europa.

¿Y en España? A mediados de los ochenta, el Ministerio de Administraciones Públicas comenzó el desarrollo de una Metodología Española (a imagen y semejanza de la francesa y la británica), que recibió el nombre de Métrica. Participaron diferentes Organismos de la Administración, y se encargó su realización final a una consultora (Coopers & Lybrand, que por entonces no estaba aún fusionada con Price Waterhouse, ni mucho menos con IBM, como ahora lo está). Se encuentra muy poca documentación de Métrica en Internet, y la poca que hay es de la versión 3, ya de fines de los noventa aquí[2].

Métrica se parecía, inicialmente, a SSADM, pero se le aligeró de bastantes pasos burocráticos, para adaptarlo más a *nuestra mentalidad*, aunque se añadieron algunos otros. Resultó una Metodología moderna y muy bien fundada. En cuanto a técnicas, básicamente usaba las mismas que los demás, ERD, DFD, Diseño Estructurado… A la versión 1 siguió la versión 2… y, por fin, la actual versión 3, desarrollada en la segunda mitad de los noventa, que incorpora ya las últimas novedades (Orientación a Objetos, UML, etc).

1.2.3. Otras metodologías

En los años 90 surgieron nuevos paradigmas, formas nuevas de concebir el análisis y la implementación de programas y bases de datos, tales como la orientación a objetos, lo que cambió en gran medida la perspectiva del desarrollo y, por extensión, la concepción de todo el ciclo del proyecto software. El Análisis y Diseño Orientado a Objetos se verá en el tema siguiente.

[2] http://www.csi.map.es/csi/metrica3/index.html

2. CICLO DE VIDA

2.1. Concepto

Se define por **ciclo de vida del software** a las distintas fases por las que pasa una aplicación software a lo largo del tiempo, desde la fase de estudio y concepción hasta la de realización, explotación y mantenimiento. El Ciclo de vida define también el modo en que esas fases se organizan entre sí.

Diversas organizaciones profesionales (IEEE o ISO/IEC) han publicado estudios relativos al ciclo de vida del software y el desarrollo de procesos. La norma IEEE 1074 y la norma ISO 12207-1.

Ambas normas dividen el proceso software en actividades, consideran una actividad como un conjunto de tareas y una tarea como una acción que transforma entradas en salidas.

El ciclo de vida abarca, toda la vida del sistema desde su concepción hasta que deja de utilizarse, por eso, a veces, se habla de **ciclo de desarrollo** como el subconjunto del anterior que empieza en el análisis y finaliza con la entrega del sistema al usuario.

2.2. PROCESOS DEL CICLO DE VIDA SOFTWARE en ISO/IEC TR 15504-2

A modo de ejemplo, vamos a ver una de las dos normas anteriores, en concreto una extensión de la ISO 12207 – 1 que es la ISO/IEC TR 15594-2.

La norma establece las **actividades** a realizar durante el ciclo de vida software, **agrupadas en procesos** según la figura, indicando que la norma **no fomenta ni especifica ningún modelo concreto de ciclo de vida, gestión de software o método de ingeniería ni establece cómo realizar ninguna de las actividades.**

Dichas actividades se agrupan en grupos de **procesos principales, procesos de soporte y cuatro procesos de la organización.**

A continuación se analizan cada uno de estos procesos para posteriormente resumir los principales modelos de ciclo de vida.

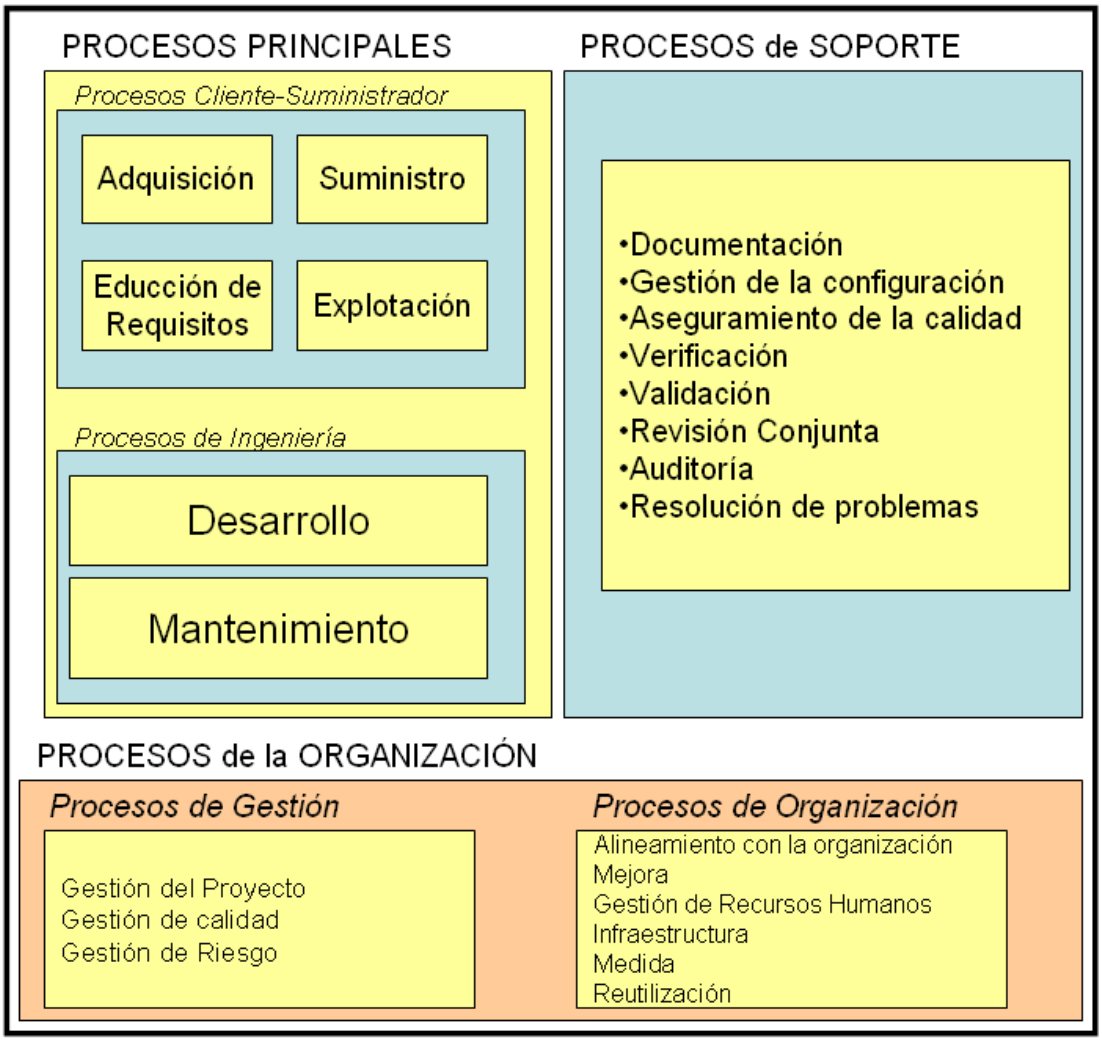


Ilustración 10: Fases ISO TR 15504-2

2.2.1. 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.

→ Procesos Cliente-Suministrador

1. Proceso de Adquisición

Son las actividades y **tareas que el comprador, cliente o usuario realiza** para adquirir un sistema, un producto o un servicio software:

- ✓ Establecer necesidades y objetivos de la adquisición
- ✓ Selección del suministrador: ¿Quién o qué empresa de informática va a suministrarme el producto que necesito?
- ✓ Control de las actividades del suministrador.
- ✓ Aceptación del producto entregado por el suministrador.

2. Proceso de Suministro

- ✓ Son las actividades y **tareas que el suministrador realiza**. El propósito es proporcionar al cliente un producto **que cumpla con los requisitos** acordados.

3. Obtención o educción de requisitos

Reunir, procesar y seguir la evolución de las necesidades y requisitos del cliente a lo largo de la vida del producto.

4. Proceso de Explotación

- ✓ Desarrollo de los planes del proyecto
    - o Ejecución de dichos planes
    - o Entrega del producto al comprador

→ Procesos de Ingeniería

Tradicionalmente, el proceso de desarrollo se ha distribuido por fases como indica la figura:

En el estándar Actualmente, en proceso de desarrollo tradicional desglosa sus fases en **actividades**, siendo las más principales las siguientes:

- ✓ Desarrollo:
    - o Análisis y diseño de los requisitos del sistema. Establece los requisitos del sistema y su arquitectura: Hardware, SSOO, …
    - o Análisis de requisitos de software: ¿qué tiene que hacer el producto?
    - o Diseño del software .
    - o Construcción o codificación del software. Además se debe verificar que lo codificado cumple con el diseño.
    - o Integración del software. Combina elementos produciendo elementos de software integrados. Deben ser igualmente verificados.
    - o Prueba del software. Probar que el producto cumple los requisitos software
    - o Integración y prueba del sistema. El sistema completo debe satisfacer las expectativas del cliente expresadas en los requisitos del sistema.

- ✓ Mantenimiento del sistema y del software. Se gestiona la modificación, migración y retirada de componentes del sistema en respuesta a nuevas peticiones del Cliente. Puede ser por necesidades de mejora o adaptación.

2.2.2. PROCESOS DE SOPORTE

Son procesos de apoyo al resto de los procesos (incluidos los mismos de soporte) y se aplican en cualquier punto del ciclo de vida del software. Son los siguientes:

→ Proceso de Documentación

Registra la información producida por un proceso o actividad del ciclo de vida. El proceso permite planificar, diseñar, desarrollar, producir, editar, distribuir, y mantener los documentos necesarios para todas las personas involucradas en el software.

→ Proceso de Gestión de la Configuración

Aplica procedimientos administrativos y técnicos durante todo el ciclo de vida del sistema para:

- ✓ Identificar, definir y establecer la línea base de los elementos de configuración del software del sistema.
- ✓ Controlar las modificaciones y las versiones de los elementos.
- ✓ Registrar el estado de los elementos y las peticiones de modificación.
- ✓ Asegurar la complexión, la consistencia y la corrección de los elementos.
- ✓ Controlar el almacenamiento, la manipulación y la entrega de los elementos.

→ Proceso de Aseguramiento de la Calidad

Garantiza que los procesos y los productos software del ciclo de vida cumplen los requisitos especificados y cumplen con los planes establecidos.

→ Proceso de Verificación

Se realiza por fases. Inicialmente por ejemplo, comprueba si los requisitos de un sistema o del software están completos y son correctos. Posteriormente comprueba si los productos software de cada fase del ciclo de vida cumplen los requisitos impuestos sobre ellos, si el diseño es consistente con el análisis, si el código lo es respecto al diseño, etc.

→ Proceso de Validación

Comprueba que el producto es funcional, que se construye el producto correcto. Comprueba que los requisitos sirven para cubrir las necesidades del usuario. Es un comprobación global.

→ Proceso de Revisión Conjunta

Revisión con el Cliente para contrastar el progreso frente a los objetivos del contrato.

→ Proceso de Auditoría

Permite establecer, **de modo independiente**, en momentos predeterminados si se han cumplido los requisitos, los planes y el contrato.

→ Proceso de Resolución de Problemas

Permite analizar y eliminar los problemas (disconformidades con los requisitos o el contrato) descubiertos durante el desarrollo, la explotación, el mantenimiento u otro proceso.

2.2.3. PROCESOS DE LA ORGANIZACIÓN

Son los procesos que emplea una organización para llevar a cabo funciones tales como la gestión, la formación del personal o la mejora del proceso. Suelen llevarse a cabo en el ámbito organizativo, fuera del ámbito de proyectos y contratos específicos.

→ Procesos de Gestión

Comprende las actividades y tareas genéricas que puede emplear una organización que tenga que gestionar sus procesos:

- ✓ Control de cualquier otro proceso
- ✓ Coordinar
- ✓ Garantizar la calidad
- ✓ Gestionar los riesgos, para reducirlos.

→ Procesos de Organización

- ✓ Proceso de Alineamiento de la Organización: todos los individuos comparten una visión común de los objetivos
- ✓ Proceso de Mejora: evalúa, mide, controla y mejora cada proceso del ciclo de vida.
- ✓ Proceso de Recursos Humanos: selección y contratación del personal adecuado para cada tarea.
- ✓ Proceso de Infraestructura: establece la infraestructura necesaria para cualquier otro proceso (hardware, software, herramientas, normas, técnicas e instalaciones para el desarrollo, la explotación o el mantenimiento).
- ✓ Proceso de Reutilización: promueve y facilita la reutilización de productos existentes, para no diseñar uno nuevo ante necesidades ya planteadas y resueltas con anterioridad.

2.3. MODELOS DE CICLO DE VIDA PARA DESARROLLO ESTRUCTURADO

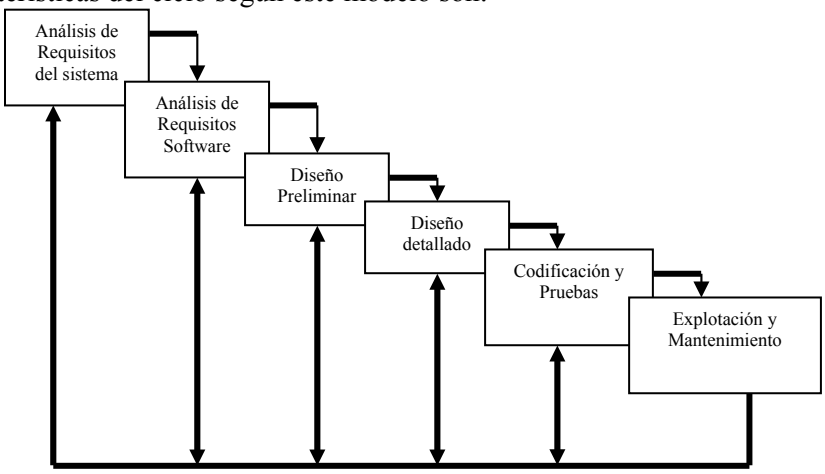
Para desarrollar el ciclo de vida se han propuesto distintos modelos de desarrollo, tanto para sistemas convencionales como para sistemas orientados al objeto. En el primer caso se considera el *modelo en cascada o Waterfall* propuesto por Royce (1970) y refinado por distintos autores (Boehm (1981), Sommerville (1985), Sigwart (1990), etc.). El *modelo incremental* (Lehman (1984) y Amescua (1995) y el *modelo en espiral* (Boehm (1988)). Por su parte, para sistemas orientados al objeto se considera el *modelo de agrupamiento o cluster* (Meyer (1990)), el *modelo fuente* (Henderson, Sellers y Edwards (1990), el *modelo remolino* (Rumbaugh (1992)) y el *modelo pinball* (Ambler (1994).

A continuación se hace una somera descripción de algunos modelos.

2.3.1. Modelo en Cascada o Tradicional

El modelo consta de un número variable de fases o etapas que se proponen para el ciclo de vida del sistema pero que generalmente suelen ser las indicadas en la figura. El inicio de cada etapa debe esperar a la finalización de la anterior.

Las características del ciclo según este modelo son:



- ✓ Cada fase empieza cuando ha terminado la fase anterior.
- ✓ Para pasar de una fase a otra es preciso conseguir todos los objetivos de la etapa previa
- ✓ Ayuda a prevenir que se sobrepasen las fechas de entrega y los costes esperados.
- ✓ Al final de cada fase, el personal técnico y el usuario tienen la oportunidad de revisar el progreso del proyecto.

Aunque es el ciclo de vida más antiguo y el más utilizado, tiene los siguientes inconvenientes:

- ✓ No refleja el proceso "real" de desarrollo del software. (Los proyectos reales raramente tienen un flujo secuencial dado que siempre hay iteraciones que pueden cubrir más de una etapa.
- ✓ Se tarda mucho tiempo en recorrer todo el ciclo, dado que hasta que no se finaliza una fase no se pasa a la siguiente, por lo que el usuario debe esperar hasta el final del proyecto para ver si el producto se ajusta o no a lo solicitado.

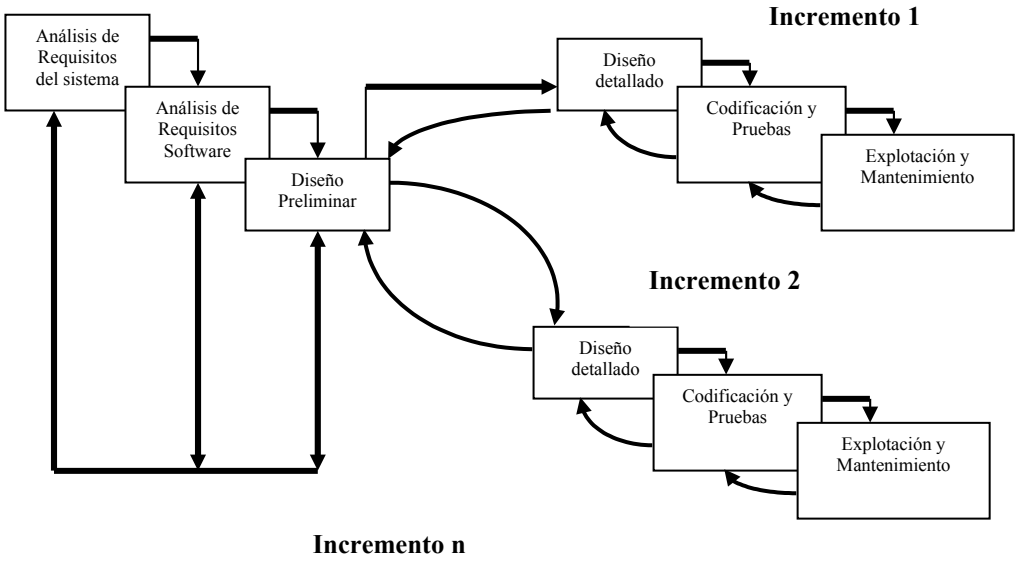
2.3.2. Modelo Incremental

Corrige la necesidad de una secuencia no lineal de pasos de desarrollo. Según dicho modelo, 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 (esto es, **cada versión o refinamiento parte de una versión previa a la que le añade nuevas funciones**).

Es un modelo que se ajusta a entornos de alta incertidumbre, por no poseer un conjunto exhaustivo de requisitos, especificaciones, diseños, etc., al comenzar el sistema, ampliándose éstos en cada refinamiento.

Representa una mejora sobre el modelo en cascada y, aunque permite el cambio continuo de requisitos, no se puede determinar si los requisitos propuestos son válidos, por lo que los errores de detectan tarde y su corrección resulta muy costosa.

Un ejemplo de modelo en cascada utilizando el desarrollo incremental es el de la figura:



2.3.3. Modelo en Espiral

El 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 de los objetivos del ciclo y considerando las restricciones, se realiza el ciclo correspondiente para, una vez finalizado, iniciar el siguiente ciclo.

- ✓ Cada ciclo en espiral comienza con los siguientes pasos:
  - o Los Objetivos de la parte del producto que está siendo elaborado.
  - o Las Alternativas principales de la implementación de esa porción del producto.
  - o Las Restricciones para cada alternativa.
- ✓ El siguiente paso es
  - o Evaluación de las diferentes Alternativas teniendo en cuenta los objetivos a conseguir y las restricciones impuestas.
  - o Si existen riesgos, formulación, de una estrategia efectiva para resolver o minimizar dichos riesgos (hacer prototipos, simulación, hacer informes de análisis de riesgos, …)
- ✓ Revisión de los resultados del análisis de riesgos
- ✓ Implementación de la fase en sí del ciclo (análoga a las vistas en el modelo tradicional e incremental en cada vuelta de la espiral). En la primera vuelta, Concepto de operación, en la segunda, Requisitos y Validación de los mismos, etc,…
- ✓ Planificación de la fase posterior

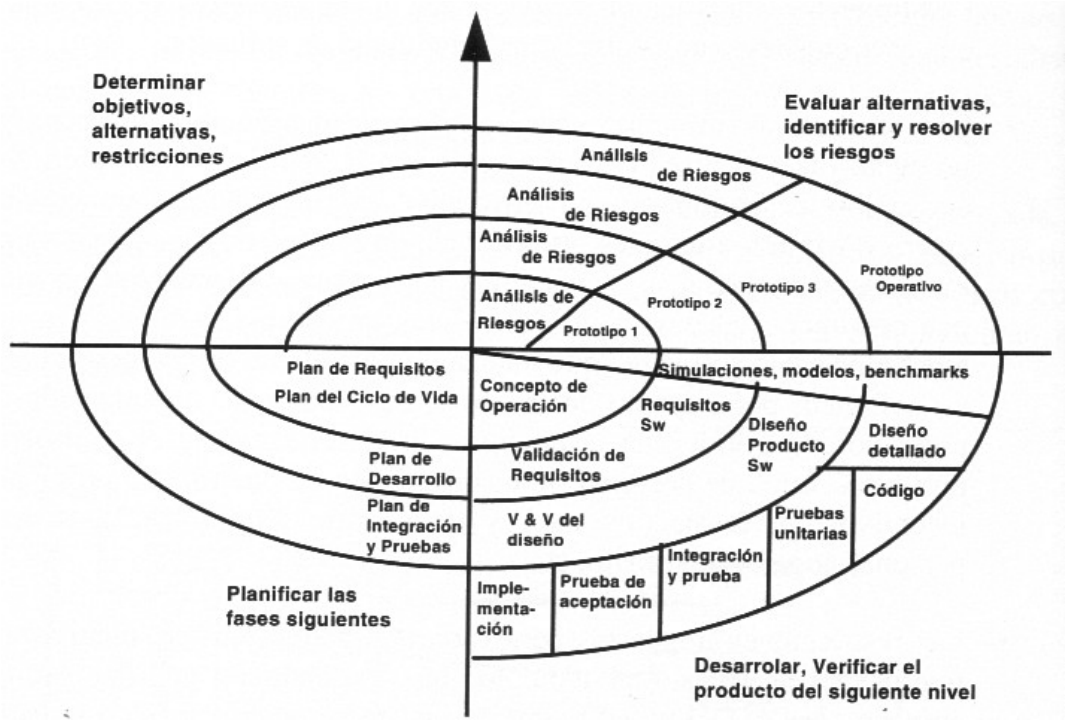


Ilustración 11: Ciclo en Espiral

Realizado el primer ciclo se volvería a empezar.

En el modelo en espiral, cada ciclo se completa con una revisión en la que participan las principales personas u organizaciones que tienen relación con el ciclo, que cubre todos los productos desarrollados en el ciclo anterior e incluye los planes para el siguiente ciclo y los recursos necesarios para llevarlo a cabo.

Estos planes para las sucesivas fases pueden incluir una partición del producto en incrementos (para desarrollos sucesivos) o en componentes (para ser desarrollados por organizaciones individuales o por personas). En este último caso pueden preverse una serie de ciclos en paralelo, uno por cada componente, añadiendo una tercera dimensión al modelo en espiral

Las principales diferencias entre el modelo en espiral y los modelos anteriores son:

- ✓ Existencia reconocida de diferentes alternativas
- ✓ Identificación de riesgos para cada alternativa . La resolución de los riesgos es realmente el centro del modelo.
- ✓ División del proyecto en ciclos, cada uno con un acuerdo al final de cada ciclo.
- ✓ El modelo se adapta a cualquier tipo de actividad.

2.4. MODELOS DE CICLO DE VIDA ORIENTADO A OBJETOS

Se verán en la Unidad de Trabajo siguiente.

Cuestión: Determina el ciclo de vida a aplicar en los siguientes casos y razona la respuesta en cada caso:

1. En una empresa para un proyecto donde está muy claro lo que se quiere implementar
2. Si el cliente es una persona inestable y de carácter voluble, y el negocio de ese cliente ha sufrido varios ajustes en el último año
3. Si el Cliente quiere supervisar los avances del proyecto y decidir sobre la marcha cada nueva implementación
4. Si el Cliente no tiene conocimientos informáticos, lo que hace complicado que se haga una idea exacta del aspecto, de los interfaces y de la funcionalidad que tendrá el producto.

3. METODOLOGÍAS DE DESARROLLO DEL SOFTWARE

3.1. DEFINICIÓN Y OBJETIVOS

Para desarrollar un proyecto de software es preciso establecer un enfoque disciplinado y sistemático del trabajo a realizar; las metodología de desarrollo a seguir se elaboran a partir del marco definido por uno o varios ciclos de vida, y deben tener determinadas características que dependerán del enfoque de desarrollo. Recordemos que el ciclo de vida elegido determina las fases y su organización, pero nada más.

No existe una definición universalmente aceptada sobre el concepto de metodología, aunque si existe un acuerdo en considerarla como "un conjunto de pasos y la especificación global y detallada de los procedimientos que deben seguirse para el desarrollo del software".

Una definición de metodología podría ser "el conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información**" (Maddison 1983).**

La metodología especifica al detalle y de modo sistemático:

- ✓ Cómo se debe dividir un proyecto en etapas.
- ✓ Qué tareas se realizan en cada etapa.
- ✓ Qué salidas se producen y cuándo se deben producir.
- ✓ Qué restricciones se aplican.
- ✓ Qué herramientas se van a utilizar.
- ✓ Cómo se gestiona y controla un proyecto.

La metodología normalmente consistirá en un conjunto de fases descompuestas en subfases (módulos, etapas, pasos, etc.) de forma que esta descomposición guíe a los desarrolladores en la elección de las técnicas que se debe elegir para cada estado del proyecto, facilitando la planificación, gestión, control y evaluación de los proyectos.

Pueden identificarse como **Necesidades** principales que se intentan cubrir con una metodología las siguientes:

- ✓ ***Mejores aplicaciones***: aunque el seguimiento de una metodología no basta para asegurar la calidad del producto final, hay además muchos pequeños factores a tener en cuenta, que se verán en unidades posteriores.

- ✓ ***Un mejor proceso de desarrollo***: que identifica las salidas de cada fase de forma que se pueda planificar y controlar el proyecto. Los sistemas se desarrollan más rápidamente y con los recursos apropiados.
- ✓ ***Un proceso estándar en la organización***: lo que aporta claros beneficios (mayor integración de los sistemas, mas facilidad en el cambio de una persona de un proyecto a otro, etc.).

Aparte de lo anteriores, las metodologías pueden tener distintos **objetivos** que hacen que difieran unas de otras. **Estos objetivos pueden ser:**

- ✓ **Registrar los requisitos** de un sistema de información de forma acertada.
- ✓ Proporcionar un método sistemático de desarrollo de forma **que se pueda controlar su progreso.**
- ✓ Construir un sistema de información **dentro de un tiempo apropiado y unos costes aceptables.**
- ✓ Construir un sistema que **esté bien documentado y que sea fácil de mantener**.
- ✓ **Ayudar a identificar** lo más pronto posible **cualquier cambio** que sea necesario realizar dentro del proceso de desarrollo.
- ✓ Proporcionar un **sistema que satisfaga** a todas las personas afectadas por el mismo, ya sean clientes, directivos, auditores o usuarios.

Al ser obligado en una metodología especificar al mayor nivel de detalle, la descomposición del proyecto llegará hasta las *tareas o actividades elementales*.

Para cada tarea se identifica **un procedimiento** que define la forma de ejecutarla y representa el marco de comunicación entre usuarios y desarrolladores. Como resultado de seguir un procedimiento, **se obtienen uno o más productos** (que pueden ser *productos intermedios* como base para realizar nuevos productos durante el desarrollo o *productos finales*). El sistema deseado está constituido por un conjunto de productos finales.

Para aplicar un procedimiento se pueden utilizar una o más **técnicas**. Suelen ser con mucha frecuencia gráficas con apoyos textuales formales y determinan el formato de los productos resultantes de cada tarea.

Para la realización de una técnica pueden utilizarse **herramientas software** que automatizan en mayor o menor grado su aplicación.

Una metodología **puede seguir uno o varios modelos de ciclo de vida**.

3.2. CLASIFICACIÓN DE LAS METODOLOGÍAS.

En función de tres dimensiones o puntos de vista (enfoque, tipo de sistema y formalidad) las metodologías pueden clasificarse en:

- ✓ ESTRUCTURADAS:
    - Orientadas a procesos
    - Orientadas a datos
        - Jerárquicos
        - No jerárquicos
    - Mixtas
- ✓ ORIENTADAS A OBJETOS

3.2.1. Metodologías de desarrollo estructurado

Sigue unos métodos de ingeniería sentando las bases para el desarrollo automatizado, Estas técnicas están dirigidas a aspectos tanto técnicos como de gestión en la construcción de software.

Su desarrollo histórico se ha fundamentado en lo siguiente:

- ✓ *Programación estructurada.* Se establecieron normas para la aplicación de estructuras de datos y de control: **Secuencia, Iteración, Selección.** Todo código debe ser hecho en base a estos tres elementos o como combinación de ellos. **Aparte se estableció el modo de subdividir un programa en subconjuntos más o menos independientes, llamados módulos**, subrutinas o subprogramas y que pueden ser funciones (al finalizar retornan un valor resultado definido) o procedimientos.
- ✓ *Diseño estructurado*. Define un nivel de abstracción utilizando el módulo como componente básico de construcción. **Se refina el concepto de modularidad**, las relaciones entre módulos y se establecen medidas de calidad de los programas. **Suelen aplicarse modelos gráficos** que definen lo que hacen los módulos del programa y cómo se relacionan: Warnier-Orr, Jackson y los Diagramas de Estructuras son ejemplos de técnicas de diseño.
- ✓ *Análisis estructurado.*

Hasta la aparición de los primeros conceptos sobre el análisis estructurado, en la gran mayoría de proyectos de desarrollo **se hacía una especificación narrativa de los requisitos** tal y como los percibía el analista. Estas especificaciones adolecían de los siguientes problemas:

- ✓ Eran **monolíticas**. Es preciso leer la especificación de los requisitos completamente parea poder entenderla
- ✓ Eran **redundantes**. Se repite frecuentemente la misma información en distintas partes del documento. Por lo que, si cambian algún requisito, el documento debe modificarse en diferentes lugares.
- ✓ Eran **ambiguas**. El informe detallado de requisitos puede ser interpretado de forma distinta por usuarios, analistas y diseñadores
- ✓ Eran **imposibles de mantener**. Cuando se llega al final del proceso de desarrollo. La especificación funcional era casi obsoleta.

Debido a estos problemas, **ha habido un movimiento gradual hacia las especificaciones funcionales:**

- ✓ **Particionadas.** De forma que se puedan leer porciones independientes de la especificación
- ✓ **Gráficas.** Compuestas por diagramas y técnicas textuales que sirven como material de referencia a la especificación

- ✓ **Mínimamente redundantes**. Afectando los cambios a una parte (módulo o grupo de módulos) de la especificación con nula o mínima repercusión en el resto. Al aislar funcionalidades, se acota el ámbito de un posible cambio de funcionalidad.

Este enfoque, conocido como *análisis estructurado o análisis descendente o análisis top-down* (parte de lo general para llegar gradualmente al máximo detalle) se utilizaba principalmente en organizaciones de desarrollo de sistemas de gestión.

Posteriormente ha sufrido los siguientes cambios:

- ✓ Diferenciar los modelos físicos y los modelos lógicos. El modelo lógico es independiente de la base de datos. El modelo físico especificará información de almacenaje propia de cada tecnología usada, el resultado del modelo físico es ejecutable en un equipo concreto.
- ✓ Ampliar las técnicas de análisis estructurado para poder modelar sistemas a tiempo real.
- ✓ Enfocarse en el modelo de datos.
- ✓ Estudiar los eventos.

La evolución histórica de las metodologías mas representativas en la Ingeniería del software se representa en la tabla adjunta:

AÑO	METODOLOGÍA
1968	*Conceptos sobre programación estructurada (DIJKSTRA)*
1974	*Técnicas de programación estructurada (WARNIER, JACKSON)*
1975	*Primeros conceptos sobre diseño estructurado (MYERS, YOURDON)*
1977	*Primeros conceptos sobre análisis estructurado (GANE , SARSON)*
1978	*Análisis estructurado (DEMARCO, WEINBERG)*
1978	*Metodología MERISE*
1981	*Versión inicial de la Metodología SSADM*
1981	*Versión inicial de INFORMATION ENGINEERING (James Martin)*
1985	*Análisis y diseño estructurado para sistemas en tiempo real (WARD, MELLOR)*
1986	*SSADM (Versión 3)*
1987	*Análisis y diseño estructurado para sistemas en tiempo real (HATLEY, PIRHBAY)*
1989	*MÉTRICA (Versión inicial)*
1990	*SSADM (Versión 4)*
1993	*MÉTRICA (Versión 2)*
1995	*MÉTRICA (Versión 2.1)*

→ Metodologías orientadas a procesos

La ingeniería del software está fundamentada en el modelo básico de **entrada → proceso → salida** de un sistema. Los datos se introducen en el sistema y el sistema responde ante ellos transformándolos para obtener las salidas.

Las metodologías estructuradas se basan en la utilización de un método descendente de descomposición funcional para definir los requisitos del sistema. Se apoyan en técnicas gráficas dando lugar a un nuevo concepto: especificación estructurada

Una **especificación estructurada** es un modelo gráfico, particionado, descendente y jerárquico de los procesos del sistema y de los datos utilizados por los procesos.

Se compone de:

- ✓ **Diagramas de flujos de datos (DFD):** representan los procesos (funciones) que debe llevar a cabo un sistema a distintos niveles de abstracción y los datos que fluyen entre las funciones. Los procesos más complejos se descomponen en nuevos diagramas hasta llegar a procesos sencillos.

- ✓ **Diccionario de datos:** es el conjunto de definiciones de todos los datos que aparecen en el DFD, tanto almacenados como en los flujos de datos.
- ✓ **Especificaciones de proceso:** describen con más detalle lo que ocurre dentro de un proceso.

1) Como **Ejemplos** de metodologías estructuradas pueden considerarse las Metodologías de DeMarco, la Metodología de Gane y Sarson y la Metodología de Yourdon/Constantine:

Todas se basan en el uso de DFDs y en concreto ésta última (Yourdon) consta de las siguientes fases:

1) Realizar los DFD del sistema.
2) Realizar el *diagrama de estructuras*, obteniéndolo a partir de los DFD's mediante dos técnicas, al *análisis de transformación* y el *análisis de transacción*.
3) Evaluación del diseño, midiendo la calidad de la estructura resultante mediante los parámetros de *acoplamiento* y la *cohesión*.
4) Preparación del diseño para la implantación dividiéndolo en unidades físicas de implantación (*cuadernos de carga*).

→ Metodologías orientada a datos no jerárquicos

Las metodologías basadas en la información se centran en la creencia de que **los datos** (tipos de datos) **constituyen el corazón del sistema de información** ya son más estables que los proceso que actúan sobre ellos.

La metodología que identifique con éxito la naturaleza de los datos de una organización partirá de una base estable para construir los sistemas de información. Los procesos también se estudian en este tipo de metodologías, pero vienen derivados de una definición inicial de los datos (modelo de datos) utilizados por la organización.

Este modelo de datos está constituido por el conjunto de entidades de datos básicas y las interrelaciones entre ellas.

Con este enfoque, todos los sistemas construidos están integrados sobre el mismo modelo de datos.

Como ejemplo de esta metodología puede considerarse la **Ingeniería de la Información** de James Martin dividida en cuatro etapas:

1) ***Planificación:*** construir una arquitectura de la información y una estrategia que soporte los objetivos de la organización.
2) ***Análisis***: comprender las áreas del negocio y determinar los requisitos del sistema.
3) ***Diseño:*** establecer el comportamiento del sistema deseado por el usuario y que sea alcanzable por la tecnología.
4) ***Construcción:*** construir sistemas que cumplan los tres niveles anteriores.

→ Metodologías orientadas a datos jerárquicos

Dentro del modelo básico de **entrada → proceso → salida** de un sistema, estas metodologías se orientan más a las entradas y salidas. **Se definen las estructuras de datos y a partir de estas se derivan los componentes procedimentales**. En este enfoque destaca:

- ✓ La estructura de control del programa debe ser jerárquica y se debe derivar de la estructura de datos del programa.
- ✓ El proceso de diseño consiste en definir primero la estructura de los datos de entrada y salida, mezclarlas todas en una estructura jerárquica de programa y después ordenar detalladamente la lógica procedimental par que se ajuste a esta estructura.

- ✓ El diseño lógico debe preceder y estar separado del diseño físico.

Como ejemplos de metodologías pueden considerarse, los métodos de JSP y JSD de Jackson, la construcción lógica de programas (LCP) y el desarrollo de sistemas estructurados de datos (LCS) , basado en la identificación de datos primarios y secundarios (calculados). A partir de la construcción de un diccionario de datos, el método se centra en el conjunto de actualizaciones y modificaciones necesarias para obtener la salida.

Originalmente, el LCS trabajaba con ficheros como estructuras de datos.

3.2.2. Metodologías de desarrollo Orientado al Objeto

El paradigma orientado a objetos, a diferencia del enfoque estructurado, trata los procesos y los datos de forma conjunta, esto es, modula tanto la información como el procesamiento. La orientación a objetos empieza con los lenguajes de programación orientados a objetos (LOO) . 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 adjuntaban un conjunto de operaciones (SIMULA, Smaltalk como prototipos y ADA, C++, Smaltalk y Objetive – C como lenguajes de desarrollo).

Entre las metodologías clásicas de análisis y diseño estructurado y las de orientación al objeto existe un cambio de filosofía apreciable. Mientras en las primeras se examinan los sistemas desde el punto de vista de las funciones o tareas que deben realizar (tareas que se van descomponiendo sucesivamente en otras tareas mas pequeñas y que forman los bloques o módulos de las aplicaciones), en la orientación al objeto cobra mucha mas importancia el aspecto de "modelado" del sistema, examinando el dominio del problema como un conjunto de objetos que interactúan entre sí.

En las metodologías tradicionales se produce una dicotomía entre los dos elementos constituyentes de un sistema: las funciones que llevan a cabo los programas y los datos que se almacenan en ficheros o en bases de datos.

Sin embargo, la orientación al objeto intenta obtener un enfoque unificador de ambos aspectos, que se encapsulan en los objetos.

Se profundizará en unidades posteriores.

3.3. CARACTERÍSTICAS DESEABLES EN LAS METODOLOGÍAS.

Una metodología debería incluir una serie de características deseables, entre las que se destacan:

1) ***Existencia de reglas predefinidas:*** que definan sus fases, tareas, productos intermedios, técnicas, herramientas, ayudas al desarrollo y formatos de documentación estándar.
2) ***Cobertura total del ciclo de desarrollo:*** pasos que hay que realizar desde el planteamiento de un sistema hasta su mantenimiento, proporcionando mecanismos para integrar los resultados de una fase a la siguiente, de forma que se pueda referenciar a fases previas y comprobar el trabajo realizado.
3) ***Verificaciones intermedias:*** sobre productos generados en cada fase para comprobar su corrección. Por medio de revisiones software que detectan inconsistencias, inexactitudes o cualquier otro tipo de defecto que se genera durante el proceso de desarrollo evitando que salgan a relucir en la fase de pruebas o en las pruebas de aceptación o durante la fase de mantenimiento.
4) ***Planificación y control:*** una forma de desarrollar software de manera planificada y controlada para que no se disparen los costes ni se amplíen los tiempos de entrega. También debería incorporar alguna técnica de control de proyectos.
5) ***Comunicación efectiva:*** entre los desarrolladores para facilitar el trabajo en grupo y con los usuarios.

6) ***Utilización sobre un abanico amplio de proyectos:*** debe ser flexible. No se deberían utilizar metodologías diferentes para cada proyecto.

7) ***Fácil formación:*** los desarrolladores deben comprender las técnicas y los procedimientos de gestión. La organización debe formar a su personal en todos los procedimientos definidos por la metodología

8) ***Herramientas CASE:*** debe estar soportada por herramientas automatizadas que mejoren la productividad del equipo de desarrollo y la calidad de los productos resultantes. Como una metodología define las técnicas que hay que seguir en cada tarea, es preciso disponer de una herramienta que soporte la automatización de dichas tareas.

9) ***La metodología debe contener actividades que mejoren el proceso de desarrollo:*** es necesario disponer de datos que muestren la efectividad de la aplicación del proceso sobre un determinado producto. Definir mediciones que indiquen la calidad y el coste asociado a cada etapa del proceso.

10) ***Soporte al mantenimiento:*** el campo de reingeniería del software debería ser tomado en cuenta por las metodologías para facilitar las modificaciones sobre los sistemas existentes.

11) ***Soporte de la reutilización del software:*** las metodologías estructuradas existentes no proporcionan mecanismos para la reutilización de componentes software. Se deberían incluir procedimientos para la creación, mantenimiento y recuperación de componentes reutilizables que no se limiten sólo al código.

3.4. SISTEMAS EN TIEMPO REAL

Son sistemas muy dependientes del tiempo que procesan información más orientada al control que orientada a los datos. Se controlan y son controlados por eventos externos.

> Una definición de sistema en tiempo real podría ser: "aquel sistema que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento".

Se caracterizan porque:

- ✓ Se lleva a cabo el proceso de muchas actividades de forma simultánea (**concurrencia**).
- ✓ Se asignan prioridades a determinados procesos (algunos requieren tratamiento inmediato y otros pueden aplazarse por periodos razonables de tiempo (**batch**).
- ✓ Se interrumpe una tarea antes de que concluya, para comenzar otra de mayor prioridad.
- ✓ Existe comunicación entre tareas.
- ✓ Existe acceso simultáneo a datos comunes, tanto en memoria como en almacenamiento secundario.

No deben confundirse los sistemas en tiempo real con los sistemas en línea (interactivos u on-line). Estos últimos interactúan con las personas, mientras que los primeros interactúan tanto con las personas como con los dispositivos físicos del mundo exterior. Si un sistema en tiempo real no responde con la suficiente rapidez, el ambiente puede quedar fuera de control, perdiendo datos de entrada. En un sistema en línea no se pierden datos, sólo "hay que esperar un poco más".

Para especificar los requisitos de estos sistemas hay que incluir nuevos conceptos para :

- ✓ El manejo de interrupciones.
- ✓ La comunicación y sincronización entre tareas.
- ✓ Gestionar procesos concurrentes.
- ✓ Dar respuesta oportuna y a tiempo ante eventos externos.

- ✓ Datos continuos o discretos.

3.5. PRINCIPALES METODOLOGÍAS DE DESARROLLO.

Distintos países han promovido el desarrollo de metodologías para unificar la forma de desarrollar sus sistemas de información. Las más conocidas son la francesa MERISE, la inglesa SSADM y la española MÉTRICA.

3.5.1. Metodología MERISE

Las mayores aportaciones de la metodología son:

- ✓ Un ciclo de vida más largo que se materializa en un conjunto definido por etapas, entre las se incluye una etapa de planificación previa al desarrollo (**esquema director**).
- ✓ Introducción de dos ciclos complementarios: ciclo de **abstracción** y ciclo de **decisión**. El primero se basa en la percepción de tres niveles de abstracción: conceptual, organizativo y físico u operativo. Se definen con dos modelos para cada nivel, un modelo de datos y un modelo de tratamientos:
    - o **Nivel conceptual**: corresponde a las finalidades de la empresa. Se ocupa de definir el ***Qué*** a través de un conjunto de reglas de gestión, objetivos y limitaciones que pesan sobre la empresa.
    - o **Nivel organizativo**: corresponde a la organización adecuada que hay que implantar para alcanzar los objetivos asignados al sistema. Requiere una especificación de los puestos de trabajo, cronología de las operaciones y la elección de la automatización así como la distribución geográfica de datos y tratamientos.
    - o **Nivel físico**: corresponde a la integración de los medios técnicos necesarios para el proyecto. Se expresan en términos de materiales o componentes software. Este nivel es el que está más sujeto a cambios como resultado de los progresos tecnológicos.

NIVELES	DATOS	TRATAMIENTOS
CONCEPTUAL	Modelo Conceptual de datos	Modelo Conceptual de Tratamientos
ORGANIZATIVO	Modelo Lógico de Datos	Modelo Organizativo de Tratamientos
FÍSICO	Modelo Físico de Datos	Modelo Operativo de Tratamientos

Las **Fases de la Metodología** son las siguientes:

- ✓ **Estudio preliminar**: analiza la situación existente para proponer una solución global atendiendo a los criterios de gestión, de la organización y decisiones adoptadas por quien tiene capacidad para ello.
- ✓ **Estudio detallado**: define, en el ámbito funcional, la solución a implementar.
- ✓ **Implementación**: realiza los programas que corresponden a las especificaciones detalladas. Se divide en dos partes:
    - o ***Estudio técnico***. Distribuye los datos en ficheros físicos y los tratamientos en módulos de programas
    - o ***Producción de programas***. Codifica y verifica los programas mediante juegos de pruebas
- ✓ **Realización y puesta en marcha**: implantación de los medio técnicos (instalación de los materiales, iniciación, captura de datos, etc.) y organizativos (formación del personal, lanzamiento de la aplicación, etc.) y recepción definitiva por parte del usuario.

3.5.2. Metodología SSADM

Los aspectos claves de esta metodología son:

- ✓ Énfasis en los usuarios: sus requisitos y participación.
- ✓ Definición del proceso de producción.
- ✓ Tres puntos de vista: datos, eventos y procesos.
- ✓ Máxima flexibilidad en herramientas y técnicas de implementación.

SSADM proporciona un conjunto de procedimientos para llevar a cabo el análisis y diseño, pero no cubre aspectos como la planificación estratégica ni entra en la construcción del código.

3.5.3. Metodología MÉTRICA

Puede consultarse en: http://www.csi.map.es/csi/metrica3/index.html

Ejercicio 1

Consulta la página de especificaciones de Métrica v3 y contesta las preguntas:

a) ¿Cómo se estructura? Haz una lista de las fases y subfases de Métrica. Lista los productos resultantes de cada fase.

b) ¿Podrías indicar si el ciclo de vida encaja en alguno de los vistos en teoría?

c) ¿Qué estándares se tuvieron en cuenta para elaborar Métrica?

d) ¿Qué tipo de metodologías soporta?

e) ¿Qué técnicas de las vistas en teoría se usan para el análisis estructurado se usan?¿Y para el diseño estructurado?¿En qué actividades?

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.

La metodología MÉTRICA detalla, para cada tarea, los participantes que intervienen, los productos de entrada y de salida así como las técnicas y prácticas a emplear para su obtención. Algunos productos son productos finales, y otros, productos intermedios que servirán de base para la realización de tareas posteriores.

Contiene referencias específicas en cuanto a seguridad y gestión de proyectos. La metodología MÉTRICA Versión 3 cubre además distintos tipos de desarrollo: estructurado y orientado a objetos, facilitando a través de interfaces la realización de los procesos de apoyo u organizativos: Gestión de Proyectos, Gestión de Configuración, Aseguramiento de Calidad y Seguridad.

Sus técnicas están soportadas por una amplia variedad de herramientas de ayuda al desarrollo disponibles en el mercado.

Para el diseño e la metodología Métrica se tuvo en cuenta el estándar ISO 12207 (ampliado en el ISO TR 15504-2) entre otros, además de las metodologías Merise y SSADM.

Tiene un enfoque orientado al proceso.

→ Ciclo de vida y productos

Como se ha dicho, la metodología descompone cada uno de los procesos en actividades, y éstas a su vez en tareas.

El orden asignado a las actividades no debe interpretarse como secuencia en su realización, ya que éstas pueden realizare en orden diferente a su numeración o bien en paralelo, como se

muestra en los gráficos de cada proceso. 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. Genera como salidas el "Catálogo de requisitos de PSI" y la "Arquitectura de información" (compuesta por el Modelo de información., Modelo de sistemas de información., Arquitectura tecnológica., Plan de proyectos., Plan de mantenimiento del PSI).
- ✓ DESARROLLO DE SISTEMAS DE INFORMACIÓN. Las actividades y tareas propuestas por la norma se encuentran más en la línea de un desarrollo clásico, separando datos y procesos. se han abordado los dos tipos de desarrollo: estructurado y orientado a objeto, donde cuenta con la mayoría de las técnicas que contempla UML 1.2. Éste proceso se subdivide en:
    - o ESTUDIO DE VIABILIDAD DEL SISTEMA (EVS). Tiene en cuenta aspectos tácticos y relacionados con aspectos económicos, técnicos, legales y operativos y constituirá la base para decidir si se sigue adelante o se abandona. Valora la inversión y los riesgos.
    - o ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI). El propósito de este proceso es conseguir la especificación detallada del sistema de información. Las técnicas usadas son el modelo entidad relación extendido y los DFD. Productos:
        - Descripción general del entorno tecnológico.
        - Glosario de términos.
        - Catálogo de normas.
        - Catálogo de requisitos.
        - Especificación de interfaz de usuario.
        - Plan de migración y carga inicial de datos.
        - Contexto del sistema.
        - Matriz de procesos/localización geográfica.
        - Descripción de interfaz con otros sistemas.
        - Modelo de procesos.
        - Modelo lógico de datos normalizado.

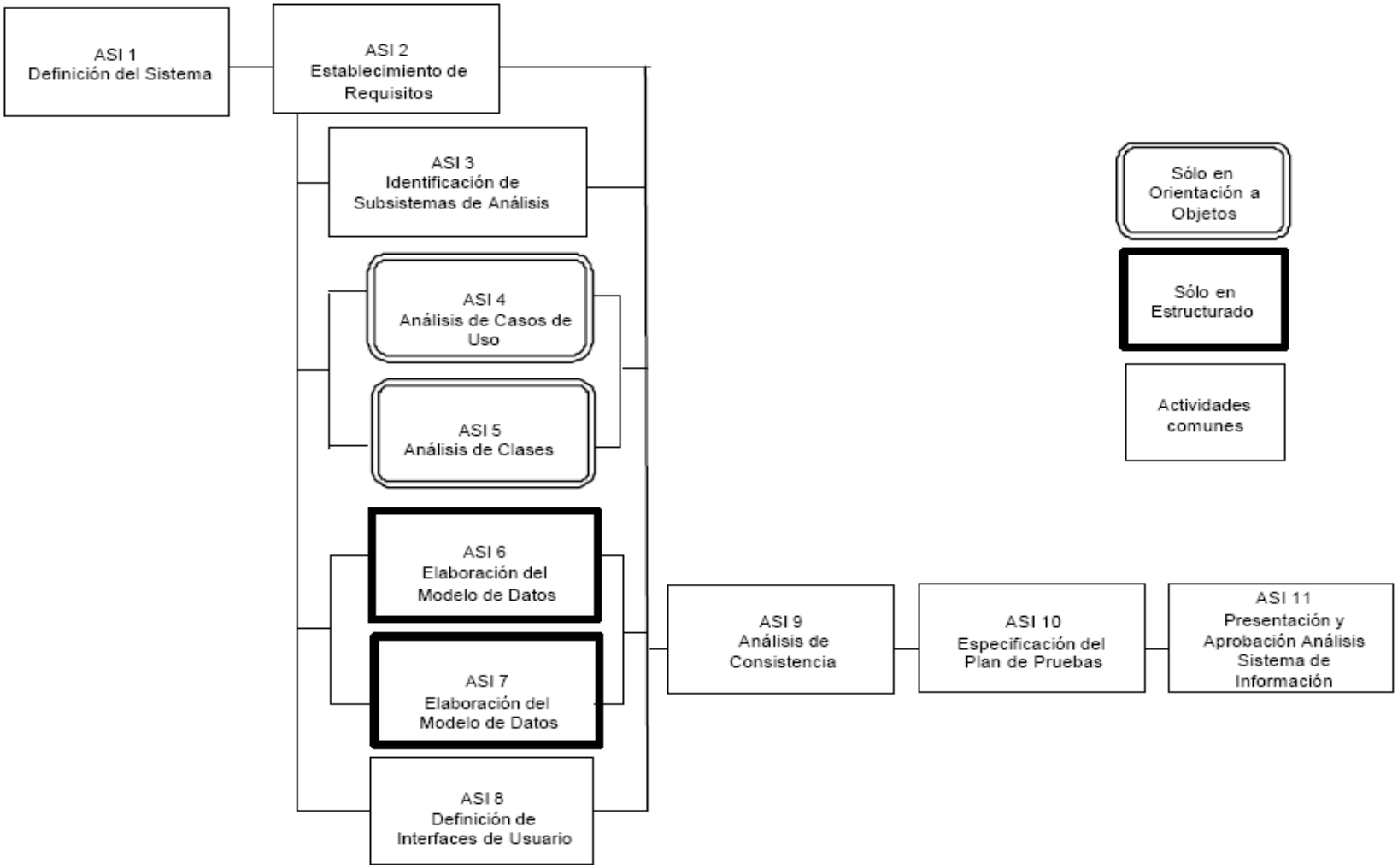

ASI 1
Definición del Sistema
ASI 2
Establecimiento de Requisitos
ASI 3
Identificación de Subsistemas de Análisis
ASI 4
Análisis de Casos de Uso
ASI 5
Análisis de Clases
ASI 6
Elaboración del Modelo de Datos
ASI 7
Elaboración del Modelo de Datos
ASI 8
Definición de Interfaces de Usuario
ASI 9
Análisis de Consistencia
ASI 10
Especificación del Plan de Pruebas
ASI 11
Presentación y Aprobación Análisis Sistema de Información
Sólo en Orientación a Objetos
Sólo en Estructurado
Actividades comunes

Ilustración 12: Actividades del proceso de Análisis

- DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI). Su propósito es obtener la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información. De este primer bloque de actividades se obtienen los siguientes productos:
  - Catálogo de requisitos (se completa).
  - Catálogo de excepciones.
  - Catálogo de normas para el diseño y construcción.
  - Diseño de la arquitectura del sistema.
  - Entorno tecnológico del sistema.
  - Procedimientos de operación y administración del sistema.
  - Procedimientos de seguridad y control de acceso.
  - Diseño detallado de los subsistemas de soporte.
  - Modelo físico de datos optimizado.
  - Asignación de esquemas físicos de datos a nodos.
  - Diseño de la arquitectura modular
  - Diseño de interfaz de usuario
- CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI). Construcción y prueba de los distintos componentes del sistema de información, a partir del conjunto de especificaciones lógicas y físicas del mismo, obtenido en el Proceso de Diseño del Sistema de Información. Como resultado de dicho proceso se obtiene:
  - Resultado de las pruebas unitarias.
  - Evaluación del resultado de las pruebas de integración.
  - Evaluación del resultado de las pruebas del sistema.
  - Producto software:
  - Código fuente de los componentes.
  - Procedimientos de operación y administración del sistema.
  - Procedimientos de seguridad y control de acceso.
  - Manuales de usuario.
  - Especificación de la formación a usuarios finales.
  - Código fuente de los componentes de migración y carga inicial de datos.
  - Procedimientos de migración y carga inicial de datos.
  - Evaluación del resultado de las pruebas de migración y carga inicial de datos.
- IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA (IAS). Este proceso tiene como objetivo principal, la entrega y aceptación del sistema en su totalidad, que puede comprender varios sistemas de información desarrollados de manera independiente, según se haya establecido en el proceso de Estudio de

Viabilidad del Sistema (EVS), y un segundo objetivo que es llevar a cabo las actividades oportunas para el paso a producción del sistema

- ✓ MANTENIMIENTO DE SISTEMAS DE INFORMACIÓN. Métrica refleja los aspectos del Mantenimiento, correctivo y evolutivo, que tienen relación con el Proceso de Desarrollo, no contemplando tareas de modificación o retirada de todos los otros componentes de un sistema de información (hardware, software de base, operaciones manuales, redes, etc.). El objetivo de este proceso es la obtención de una nueva versión de un sistema de información desarrollado con MÉTRICA, a partir de las peticiones de mantenimiento que los usuarios realizan con motivo de un problema detectado en el sistema o por la necesidad de una mejora del mismo. Los productos que se obtienen en este proceso son los siguientes:
    - Catálogo de peticiones de cambio.
    - Resultado del estudio de la petición.
    - Propuesta de solución.
    - Análisis de impacto de los cambios.
    - Plan de acción para la modificación.
    - Plan de pruebas de regresión.
    - Evaluación del cambio.
    - Evaluación del resultado de las pruebas de regresión

Existen otras tareas que la metodología denomina, de modo un tanto confuso Interfaces. Estas son tareas realizadas paralelamente a las anteriores como soporte o gestión:

Las "interfaces" descritas en la metodología son:

- ✓ Gestión de Proyectos (GP)
    - Actividades de Inicio del Proyecto (GPI), que permiten estimar el esfuerzo y establecer la planificación del proyecto.
    - Actividades de Seguimiento y Control (GPS), supervisando la realización de las tareas por parte del equipo de proyecto y gestionando las incidencias y cambios en los requisitos que puedan presentarse y afectar a la planificación del proyecto.
    - Actividades de Finalización del Proyecto, cierre y registro de la documentación de gestión.
- ✓ Seguridad (SEG). Análisis de Riesgos.
- ✓ Aseguramiento de la Calidad (CAL)
- ✓ Gestión de la Configuración (GC): su función es identificar, definir, proporcionar información y controlar los cambios en la configuración del sistema, así como las modificaciones y versiones de los mismos.

4. Bibliografía y Enlaces

- ✓ Análisis y Diseño Detallado de Aplicaciones informáticas de Gestión. Mario Piattini; Calvo-Manzano; Cervera; Fernández. Ed. RA-MA.
- ✓ http://eltamiz.com/elcedazo/series/historia-de-un-viejo-informatico/
- ✓ http://www.csi.map.es/csi/metrica3/index.html