Herramientas y Metodologías de

Análisis y Diseño Estructurado

Apunte de la Cátedra Metodologías de Desarrollo de Software I

Claudia Marcos – Edgardo Belloni

1 Introducción

El desarrollo de sistemas pequeños, en la cual participan una o dos personas, es una tarea simple. Los cambios naturales que surgen durante el ciclo de desarrollo del sistema no producen una gran propagación de cambios en el sistema. Sin embargo, si el sistema es grande y en su desarrollo participan varios grupos de personas desarrollando una tarea específica, hay que tener en cuenta no solo la comunicación con el usuario sino también la inter-relación entre los distintos grupos de trabajo.

Algunos de los problemas comunes que los desarrolladores encuentran en la construcción de software de cierta complejidad son los siguientes:

- El Dominio de Aplicación no es conocido.
- La Comunicación con el usuario.
- La Comunicación con el grupo de desarrollo.
- La Carencia de buena documentación.

Por esta razón, es necesario seguir una serie de pasos sistemáticos para que los diferentes grupos de desarrollo posean una buena comunicación. Estos pasos son brindados por los modelos de ciclo de vida, los cuales están constituidos por diferentes etapas:

Especificación de requerimientos: Se realizan entrevistas con el usuario identificando los requerimientos y necesidades del usuario.

Análisis: Modela los requerimientos del usuario.

Diseño: Se modela la solución del sistema, teniendo en cuenta el ambiente de implementación a utilizar, por ejemplo, si el sistema es centralizado o distribuido, la base de datos a utilizar, lenguaje de programación, performance deseada, etc.

Implementación: Dado el lenguaje de programación elegido se implementa el sistema.

Testeo: En esta etapa se verifica y valida el sistema teniendo en cuenta algunos criterios determinados por el grupo correspondiente.

Mantenimiento: Es la etapa más difícil de desarrollo del sistema, actualiza y modifica el sistema si surgen nuevos requerimientos.

Varios métodos para describir el ciclo de vida de un sistema, uno de ellos es el desarrollo estructurado en Cascada.

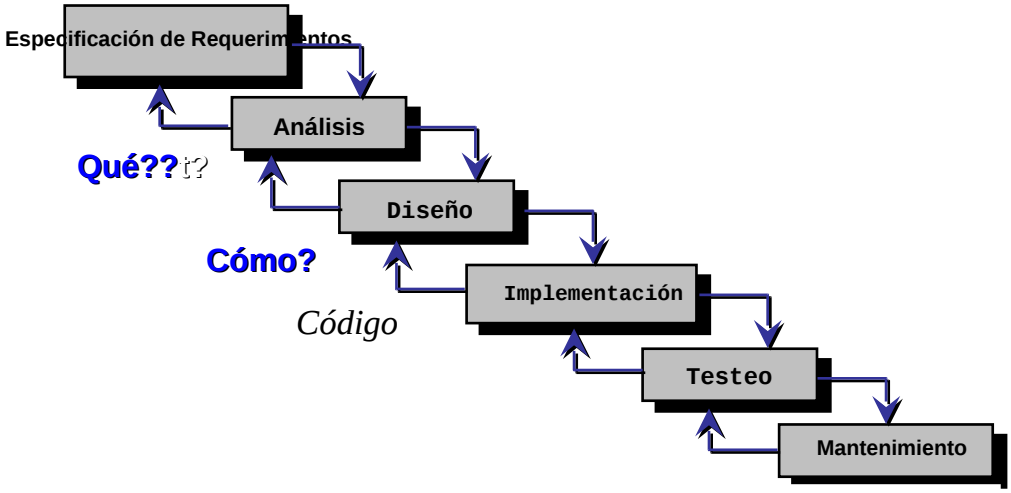

Fig. 1 Modelo de Ciclo de Vida en Cascada

En un principio fue de gran utilidad pero el problema es que para pasar de una etapa a la otra había que terminar la primera, produciendo un gran problema si algún cambio era requerido. La etapa de Mantenimiento consumía el 80% del costo de producción.

Debido a los nuevos requerimientos en el desarrollo de software, surgieron muchos otros modelos que trataban de solucionar los problemas existentes, los cuales se basaron en el modelo en Cascada. Por ejemplo, el Modelo en Espiral, en el cual es sistema se desarrolla incrementalmente (fig.2).

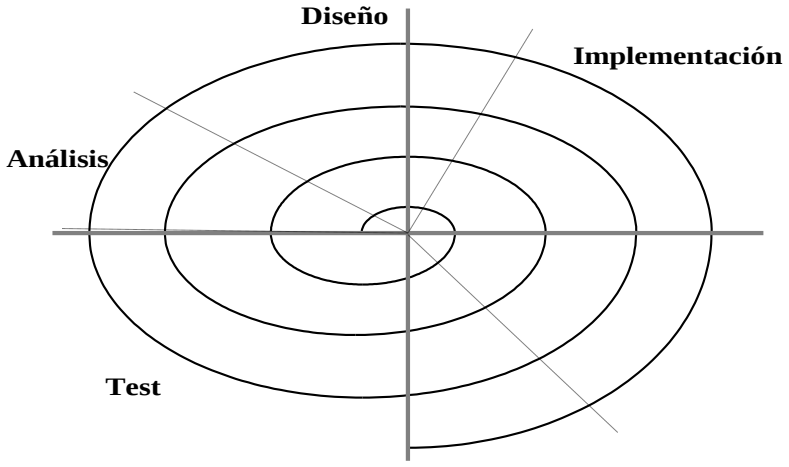

Fig. 2 Modelo de Ciclo de Vida en Espiral

Los modelos propuestos poseen básicamente las mismas etapas, pero varían básicamente en:

- los métodos y herramientas utilizadas en cada actividad,
- los controles requeridos, paralelismo en las actividades y
- en las salidas de cada etapa.

No es aconsejable elegir un modelo y seguirlo al detalle sino que se debe adaptar a las características del proyecto que esta siendo desarrollado.

Los métodos de desarrollo de software pueden dividirse en dos grupos: función/dato y orientados a objetos.

Orientado a Función/Dato	Orientado a Objetos
Enfasis en la transformación de datos	Enfasis en la abstracción de datos.
Funciones y datos tratados como entidades separadas	Funciones y datos encapsulados en entidades fuertemente relacionadas
Difícil de entender y modificar	Facilidades de mantenimniento
Funciones, usualmente, dependientes de la estructura de los datos.	Facilitates maintenance by subclassing
	Mapeo directo a entidades del mundo real

Orientado a Función/Dato: Aquellos métodos en los cuales las funciones y/o los datos son tratados como entidades independientes. Estos sistemas resultan difíciles de mantener. El mayor problema es que las funciones generalmente dependen de la estructura de los datos. A menudo diferentes tipos de datos tienen distintos formatos y se necesita verificar el tipo del dato (con sentencias IfThen o CASE), produciendo programas difíciles de leer y modificar. Si se desea hacer alguna modificación en la estructura de los datos se debe modificar en todos los lugares donde es utilizado.

Otro problema es que una persona no piensa naturalmente en términos de una estructura. La especificación de requerimientos se hace en lenguaje común, se especifica la funcionalidad que debe tener el sistema y no en cómo se deben estructurar los datos.

Orientado a Objetos: Son aquellos métodos en los que los datos y funciones están altamente relacionados. El énfasis está centrado en la abstracción de datos. Se piensa en forma natural, los objetos son mapeados a entidades del mundo real. Los programas son fácilmente mantenibles y extensibles por medio de la construcción de subclases.

Varios métodos de desarrollo de software han sido propuestos para cada uno de estos grupo, algunos de los cuales son descriptos en la figura 3.

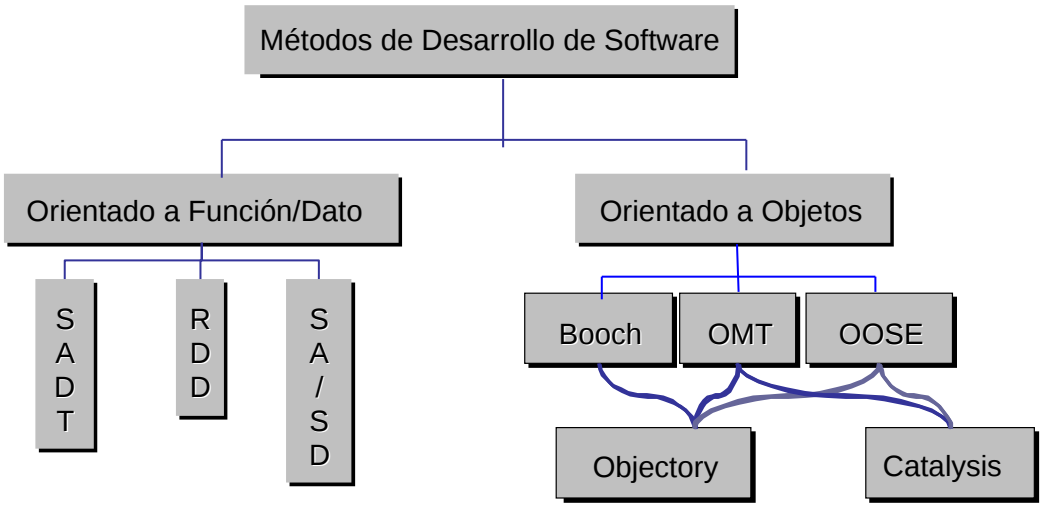

Fig. 3 Métodos de Desarrollo de Software

Donde:

SADT:	Structured Analysis and Design Technique [Ross85]
RDD:	Requirement Driven Design [Alford85]
SA/SD:	Structured Analysis and Structured Design [Yourdon&Constantine79]
OOSE:	Object-Oriented Software Engineering [Jacobson94]
OOA:	Object-Oriented Analysis [Goldberg]
OMT:	Object Modelling Technique [Rumbaugh93]
Objectory:	Objectory [Booch&Jacobson&Rumbaugh98]
Catalysis:	Catalysis [D'Souza98]

En el corriente curso: ***Metodologías de Desarrollo de Software I*** nos centraremos en los métodos de desarrollo de software orientados a función/datos y en las herramientas propuestas para el modelamiento de los diferentes aspectos de un sistema: datos, control y funciones.

ASML es una metodología de desarrollo estructurado de sistemas que cubre todo el ciclo de vida de desarrollo. Esta metodología integra las principales ideas del *Análisis Estructurado* [DeMarco 79; Gane 79] y el *Diseño Estructurado* [Constant74; Yourdon 78] en un marco conceptual único y consistente.
Si bien la definición original de la metodología puede reconocerse en los libros de Ward [Ward 83; 86] y MacMenamim [MacMenam84], le cabe una mejor definición al ser interpretada como: la conjunción del *Análisis Estructurado Moderno* [Yourdon 89] y el *Diseño Estructurado* [Constant74; Yourdon 78], con extensiones para el modelado sistemas de tiempo real [Ward 86].

2 1 Estructura de la Metodología

ASML es una metodología que integra todas las ideas involucradas en el análisis y diseño estructurado. Conjuga las técnicas y herramientas de modelado usadas en el *Análisis Estructurado Moderno y en el Diseño Estructurado* dentro de una organización que destaca los diferentes modelos necesarios para: obtener una buena comprensión del problema, y diseñar una solución de buena calidad (mantenible, adaptable, etc.). Separa el modelado de un sistema en una jerarquía de modelos necesarios para comprender diferentes propiedades del mismo. Dicha jerarquía de modelos se presenta en la figura 1.

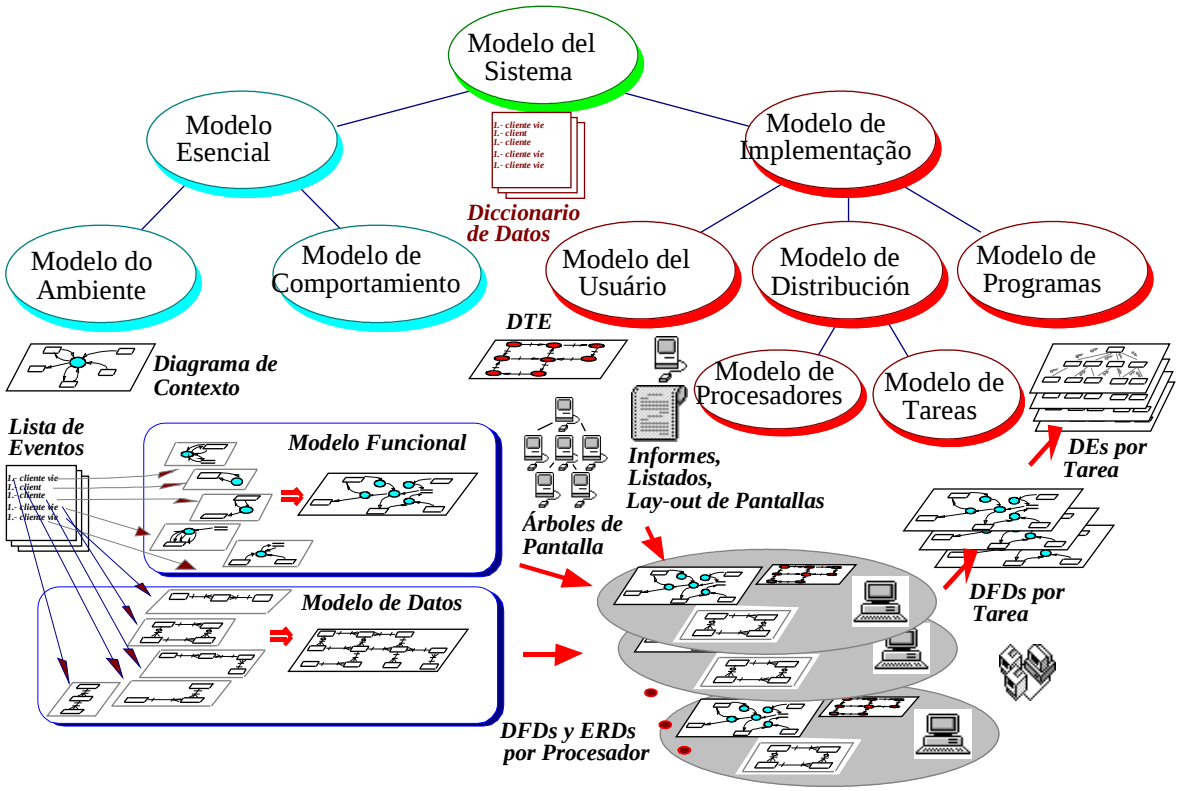

Fig 2: Jerarquía de Modelos

El *Modelo del Sistema* está dividido en dos modelos generales:

- ✓ El *Modelo Esencial* [McMenam84; Yourdon 89]: Representa la etapa de *Análisis Estructurado*. Construcción de un modelo libre de detalles tecnológicos.

- ✓ El *Modelo de Implementación* [Page 88; Ward 86; Yourdon 78; Yourdon 89]: Representa la etapa de *Diseño Estructurado*. Instanciación de un *Modelo Esencial* con una *Tecnología* dada.

2.1 2El Modelo Esencial

Puede ser considerado como la aplicación de la metodología de *Análisis Estructurado Moderno* de Yourdon. La idea fundamental con la que el modelo esencial es concebido es la de *Tecnología Perfecta* en la cual no hay restricciones de cantidad de memoria, tamaño del disco o velocidad del procesador. Dos modelos componen el modelo esencial:

- ✓ El *Modelo del Ambiente*: Creación de un *Diagrama de Contexto* y de una *Lista de Eventos*, describe los estímulos que recibe el sistema y las respuestas generadas por los estímulos.

- ✓ El *Modelo del Comportamiento*: Creación de un DFD, y un ERD por cada uno de los eventos de la *Lista de Eventos*. Los DFDs por eventos se unen en un único DFD (el *Modelo Funcional*) y los ERDs por eventos se unen en un único ERD (el *Modelo de Datos*). Se acostumbra, también, modelar el comportamiento externo del sistema con DTE, árboles de pantallas o menúes, etc. La creación simultánea del modelo de datos, modelo funcional y modelo de interfaz o comportamiento externo, ayuda en la validación y completitud del modelo esencial (descubriendo eventos no considerados por ejemplo).

Todos los criterios de modelado y, principalmente de validación, descriptos en la metodología de *Análisis Estructurado Moderno* pueden (y deben) ser aplicados en esta etapa para obtener un modelo esencial de calidad y que sea consistente.

2.2 3El Modelo de Implementación

A partir de esta etapa, el modelo esencial es instanciado en una tecnología dada. Se debe considerar ahora, las imperfecciones de la tecnología y determinar: la cantidad de procesadores necesarios, las cualidades de estos procesadores, el tamaño de disco necesario de acuerdo al volumen de la información a ser almacenada, etc. Luego se diseña la solución sobre la base de esas restricciones tecnológicas.
La creación del modelo de implementación se fundamenta en la creación de tres modelos, uno de ellos en forma independiente (el modelo del usuario o de la interfaz hombre-máquina) y los otros dos en forma encadenada en un proceso incremental de refinamiento e incorporación de detalles:

- ✓ El *Modelo del Usuario*: La interfaz hombre-máquina es modelada en todos sus detalles, estilo (árboles de menúes, lenguajes de comandos, manipulación directa, etc.), lay-out y formato de pantallas, formato de informes y listados, diseño de

pantallas para el ingreso de datos y presentación de resultados, estilo de mensajes de error, secuencialidad, etc. La creación de este modelo es independiente del resto de los modelos que conforman el de implementación, y puede ser desarrollado en paralelo. Las interfaces deben ser diseñadas para cada uno de los procesadores (del modelo de procesadores) y para cada una de las tareas (del modelo de tareas).

- ✓ El *Modelo de Distribución*: Describe todas las decisiones relativas a la arquitectura de hardware (modelo de procesadores) y a la estructuración general de la arquitectura de software (modelo de tareas). Se incorporan, en los modelos creados hasta este punto algunas *Distorsiones* destinadas a optimizar el uso de esa tecnología. El criterio fundamental es: *Minimizar todo lo posible las distorsiones agregadas*.

    - ✓ El *Modelo de Procesadores*: El modelo comportamental (modelo de datos, modelo funcional y modelo de comportamiento externo o de interfaz) es subdividido por procesadores. Se aplican criterios c*ualitativos* (por ejemplo: necesidad de monitores de alta resolución gráfica) y c*uantitativos* (por ejemplo: velocidad del procesador, volumen de información almacenada, etc.) para seleccionar los procesadores, sistemas operativos, software y hardware de red, etc. Las *distorsiones* agregadas corresponden a la partición del DFD, el DER y el DTE en procesadores, a la duplicación de procesos en varios procesadores, refinamiento de procesos y entidades o depósitos de datos (para asociar parte en un procesador y parte en otro) y a la incorporación de procesos para el control de la comunicación entre procesadores (siempre que la tecnología no solucione el problema de manera transparente).

    - ✓ *El Modelo de Tareas*: Los modelos resultantes de la creación del modelo de procesadores son estudiados por separado (un procesador por vez), para determinar tareas diferentes (que serán programas diferentes que se pueden ejecutar concurrentemente o no). La *distorsión* agregada en esta etapa representa la subdivisión del modelo funcional de un procesador (el DFD) en distintos DFDs (uno por tarea) agrupando procesos batch, interactivos o en tiempo real, partes del DFD aisladas del resto (comunicación solamente a través de depósitos de datos), etc. Además, es probable que sea necesario agregar procesos de control de concurrencia y sincronización para el acceso a recursos compartidos (como por ejemplo los depósitos de datos).

- ✓ El *Modelo de Programas*: La estructura del programa que implementa cada una de las tareas resultantes de las etapas de modelado de procesadores y tareas, es diseñada mediante la aplicación de las técnicas y estrategias descriptas por el *Diseño Estructurado* (por ejemplo: Análisis de Transformaciones y Transacciones) y mejorada con la aplicación de criterios de calidad (por ejemplo: Cohesión, Acoplamiento, etc.).

3 4 Secuencia de Creación de los Modelos

3.1