METODOLOGÍAS DE DESARROLLO DE SOFTWARE

INTRODUCCION

Definición de método y metodología

Un método es un conjunto de pasos sistemáticamente organizados para alcanzar un objetivo dado.
Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Una metodología utiliza un conjunto de métodos.
Una metodología especifica:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué salidas se producen y cuando se deben producir
- Qué restricciones deben aplicarse.
- Qué técnicas y herramientas se emplean.
- Cómo se controla y gestiona un proyecto.

La descomposición del proceso de desarrollo se realiza hasta el nivel de **tareas**. Para cada tarea se identifica un **procedimiento** que define la forma de ejecutarla. Como resultado se obtiene uno o más productos.
Para aplicar un procedimiento se puede utilizar una o más **técnicas**. Estas suelen ser con frecuencia gráficas con apoyos textuales y determinan el format de los productos resultantes de cada tarea.
Para la realización de una técnica, podemos apoyarnos en **herramientas** que automatizan en mayor o menor grado su aplicación.

Artesanía Vs. Ingeniería

Artesanía (desarrollo convencional)

- Subjetividad, se basa en la experiencia del desarrollador
- Resultados no reproducibles
- Los resultados finales son impredecibles
- No hay forma de controlar lo que está sucediendo en el Proyecto

Ingeniería

- Objetivos + Restricciones + Decisiones + Metodología
- Sistematización de las Decisiones
- Resultados reproducibles

Necesidad de Metodologías para el desarrollo de SW

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 interrelación 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 y las metodologías de desarrollo.**

Se observa que muchos desarrolladores inseguros de sus propias capacidades y experiencia, a menudo claman que muchos métodos bien probados de desarrollo, no son "prácticos" en el "mundo real". Esto se ha expresado sobre la programación estructurada, las bases de datos relacionales, la programación orientada a objetos, y muchas otras prácticas que los desarrolladores "mejor pagados" de la industria, utilizan regularmente. – Paul Conte -

Clasificación de Metodologías de Desarrollo de SW

Las metodologías de desarrollo de software pueden clasificarse en dos grandes 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 mantenimiento.
• Funciones, usualmente, dependientes de la estructura de los datos.	• 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 If-Then 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.

Dentro de esta categória de métodos se diferencian dos enfoques:

- Desarrollo basado en el flujo de información.
- Desarrollo basado en la estructura de Datos.

Desarrollo basado en el flujo de información: Conocido comunmente como enfoque estructurado (SA/SD), cuyos principales propulsores fueron DeMarco, Gane-Sarson, y Yourdon. Se caracteriza por centrar el desarrollo en el análisis de los flujos de datos y los procesos que los transforman, obteniendo una Especificación Estructurada. Su característica principal es

- Durante el análisis se genera una especificación estructurada consistente en Diagramas de Flujos de Datos, Diccionario de Datos, y Especificaciones de Procesos.
- Durante el diseño se deriva un modelo de estructura modular de los programas partiendo de los DFD analizando centros de transformación y transacción.

Desarrollo basado en la estructura de datos: Es un enfoque que deriva la estructura del sistema a partir de las estructuras de datos que deben tratarse. Los principales exponentes de este enfoque son las metodologías de Jackson y Warnier-Orr. Se caracteriza por:

- 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 las estructuras de los datos de entrada y salida, mezclarlas todas en una estructura jerárquica de programa y después ordenar detalladamente la lógica procedimental para que se ajuste a esta estructura
- El diseño lógico debe preceder y estar separado del diseño físico

Orientado a Objetos: Son aquellos métodos en los cuales datos y funciones están altamente relacionados. El énfasis está centrado en la abstracción de datos y operaciones en una única unidad llamada *objetos*. Se piensa en forma natural, los objetos son mapeados desde las entidades del mundo real. Los programas tienden a ser más fácilmente mantenibles y extensibles.
Se destacan dos enfoques distintos:

- *Revolucionario, puro u ortodoxo*: Rompen con las metodologías tradicionales. (Ejemplos: metodologías OOD de Booch, CRC/RDD de Wirfs-Brock)
- *Sintetista o evolutivo*: Toman como base los sistemas estructurados y conforman elementos de uno y otro tipo (Ejemplos: metodología OMT de Rumbourgh).

El Ciclo de Vida de Desarrollo de Software

Definición: conjunto de actividades que se llevan a cabo durante un proyecto de software.
Se suele utilizar como sinónimo al término metodología, pero este último es más abarcativo ya que el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no cómo. Esto lo debe indicar la metodología.

Objetivos del C.V.

1) *Definir las actividades* a llevarse a cabo durante un proyecto de sistemas.
2) Lograr *congruencia* entre multiples proyectos de SW.
3) Proporcionar *puntos de control administrativo* de las decisiones de continuar o no con el proyecto.

Actividades comunes del ciclo de vida de desarrollo

Independientemente del enfoque, el ciclo de desarrollo de software abarca las siguientes actividades:

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.

Estilos de Gestión del C.V.

Ciclo en Cascada

Existen varios estilos para gestionar el ciclo de vida de un sistema. El más antiguo y tradicional es el desarrollo en Cascada (fig. 1).

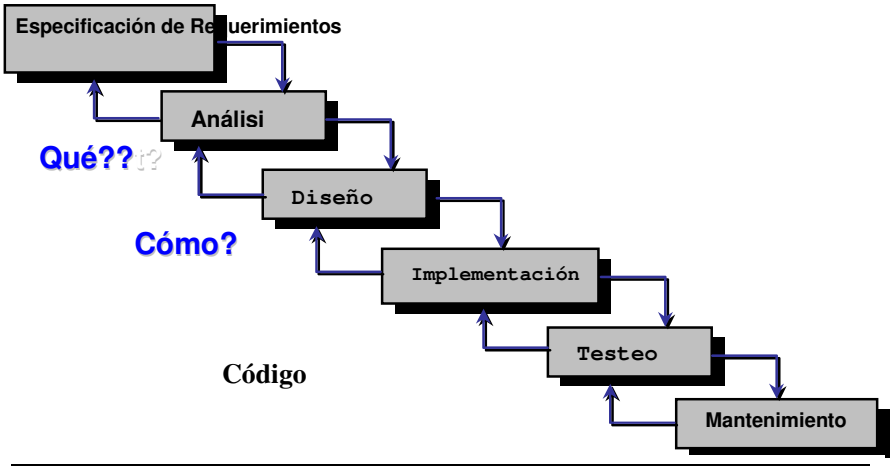


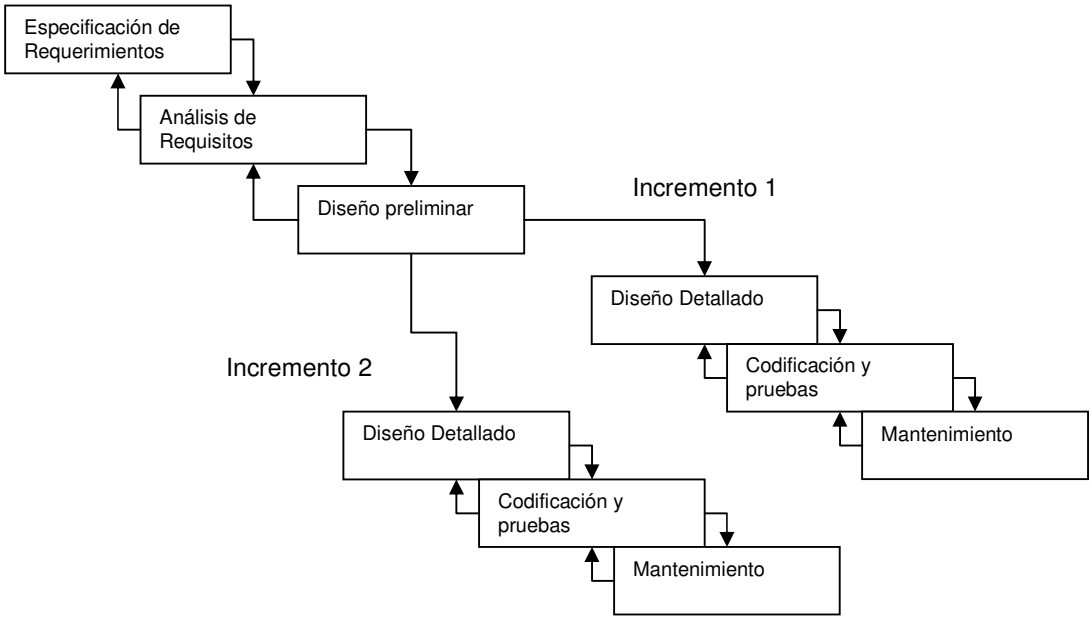
Fig. 1 Modelo de Ciclo de Vida en Cascada

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 otros modelos que trataban de solucionar los problemas existentes, los cuales se basaron en el modelo en Cascada, como el Modelo Incremental y el Modelo en Espiral.

Modelo Incremental

En este modelo, se crea 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.



Modelo en Espiral

En el Modelo en Espiral, el sistema se desarrolla incrementalmente. A diferencia de los modelos incrementales tradicionales, el modelo en espiral pone énfasis en:

- Existe un reconocimiento explícito de las diferenes alternativas para alcanzar los objetivos de un proyecto.
- La identificación de riesgos asociados con cada una de las alternativas y las diferentes maneras de resolverlos son el centro del modelo. Esto es algo no contemplado en los modelos tradicionales donde a menudo se deja lo más complejo para el final.

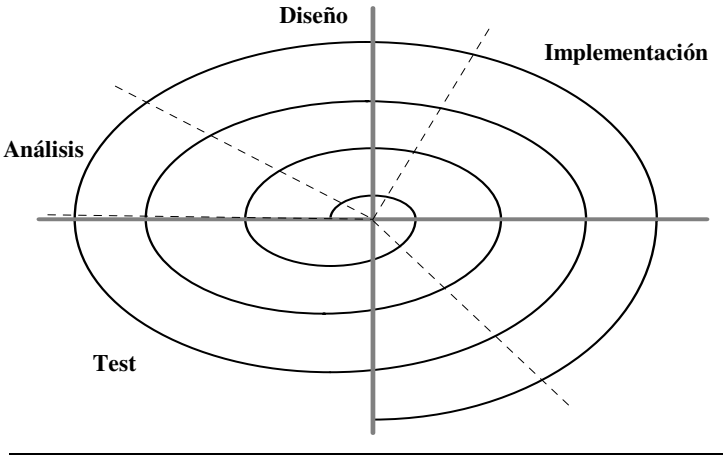


Fig. 2 Modelo de Ciclo de Vida en Espiral

Los modelos propuestos poseen básicamente las mismas etapas, pero varían 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.

Modelo de Ciclo de Vida por Prototipos

Este modelo se basa en la construcción de un prototipo el cual se desarrolla de modo incremental. A partir del prototipo desarrollado, se construye luego el sistema definitivo. Algunas de las características sobresalientes de este enfoque son:

- Desarrollo incremental con el usuario.
- Debe disponerse de herramientas modernas:
  - Generadores de pantallas
  - Generador de reportes
  - Diccionario de datos integrado
  - Lenguaje de programación de cuarta generación o generadores de código
  - Lenguajes de consulta no planeada
  - Administradores de B.D. relacionales
- El prototípo NO REEMPLAZA el sistema definitivo, el cual debe ser construído con una fase de diseño posterior.

- Candidatos:
  - El usuario no quiere modelos en papel
  - El usuario quiere determinar requerimientos por tanteo
  - Sistemas con alta interactividad
  - No se requieren complejos algorítmos
  - Son más importantes los aspectos de presentación que de cálculo
  - Sistemas de apoyo para la toma de decisiones

Enfoque Estructurado - Tipos de Ciclo de Vida de Proyecto

Tomando en consideración la evolución de las metodologías estructuradas, pueden categorizarse los ciclos de vida de desarrollo de la siguiente forma, según la introducción de las técnicas estructuradas en el proceso:

1) Clásico
2) Semiestructurado
3) Estructurado

C.V. Clásico

Históricamente el más antiguo. Se caracteriza por:

- Implementación Ascendente.
- Las fases se suceden en orden secuencial (cascada).
- Nada está hecho hasta que todo está terminado.
- Las fallas mas triviales se encuentran al comienzo y las más graves al final.
- La eliminación de fallas al final suele ser muy difícil.
- Especificación funcional
  - monolítica
  - redundante
  - ambiguas
  - imposibles de mantener
- Centrada en las funciones descuidando el diseño de los datos

C.V. Semiestructurado

A mediados de los 70 se desarrollan los métodos de programación y diseño estructurado. La aplicación de los mismos produce cambios en el estilo del ciclo de vida:

- Implementación Descendente: primero se codifican módulos de alto nivel y luego los de bajo nivel.
- Se utiliza diseño estructurado y programación estructurada
- El análisis produce una especificación de requerimientos textual narrativa, la cual debe ser convertida por el diseñador.

C.V. Estructurado

Durante los 80 maduran los métodos de análisis estructurado, permitiendo de esta forma establecer un ciclo de vida de desarrollo que abarque todas las actividades desde el enfoque estructurado. Este ciclo de vida se caracteriza por:

- Incremental. Las actividades se realizan en forma paralela en progresivos niveles de refinamiento.
- Existe retroalimentación entre actividades.
- Las fallas y desvíos se detectan y corrigen oportunamente.
- Utilización de modelos gráficos con apoyo textual.
- Especificación funcional
  - gráfica
  - particionada
  - mínimamente redundante
- Centrada en los flujos de datos y transformaciones que sufren.

El ciclo de vida estructurado puede variar según las necesidades desde un extremo radical al otro extremo conservador.

- **Implementación Radical**: Todas las actividades del C.V. se realizan simultáneamente.

- **Implementación Conservadora**: La actividad N debe completarse antes de comenzar la actividad N+1 (cascada clásica).

Factores que influyen en el tipo de implementación

- Cuán voluble es el usuario (+radical)
- Presión de entrega de resultados inmediatos (+radical)
- Presión de elaboración de presupuesto del administrador de proyecto (+conservadora)
- Peligro de cometer errores técnicos graves (+conservadora)

Evolución histórica de las metodologías

AÑO	METODOLOGÍA
1968	Conceptos sobre la programación estructurada de DIJKSTRA
1974	Técnicas de programación estructurada de WARNIER y JACKSON
1975	Primeros conceptos sobre diseño estructurado de MYERS y YOURDON
1977	Primeros conceptos sobre análisis estructurado GANE y SARSON
1978	Análisis estructurado: DEMARCO y WEINBERG
1985	Análisis y Diseño estructurado para sistemas de tiempo real de WARD y MELLOR
1987	Análisis y Diseño estructurado para sistemas de tiempo real de HATLEY y PIRHBAY
1989	Análisis Estructurado Moderno YOURDON
1990	OOA/OOD Coad/Yourdon
199x	OOD BOOCH
199x	OMT/OMT2 Rombaugh
199x	OOSE – Jacobson
1999	UML OMG
2000	RUP – Jacobson/Rumbaugh/Booch

Características Deseables de una Metodología

- Existencia de reglas predefinidas
- Cobertura total del ciclo de desarrollo
- Verificaciones intermedias
- Planificación y control
- Comunicación efectiva
- Utilización sobre un abanico amplio de proyectos
- Fácil formación
- Herramientas CASE
- Actividades que mejoren el proceso de desarrollo
- Soporte al mantenimiento
- Soporte de la reutilización de software

Referencias:

- Metodología ASML – Dra. Claudia Marcos / Ing Edgardo Belloni - UNICEN http://www.exa.unicen.edu.ar/
- "Análisis Estructurado Moderno" – Ed.Yourdon – ISBN: 9688803030
- "Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión" – M.Piattini