La gestión, los procesos y las metodologías de desarrollo de software

Sonia Pinzón* Juan Carlos Guevara Bolaños**

Resumen

£J1 presente documento proporciona una visión general de la manera como ha evolucionado el desarrollo de software, mostrando la necesidad e importancia que tiene el hecho de implementar software desde una concepción de gestión de proyectos. Adicionalmente, se aclaran los términos proceso de desarrollo de software, modelos de ciclo de vida, metodologías y métodos, debido, principalmente, a que algunos autores los utilizan indiscriminadamente, sin establecer claramente las diferencias que existen entre estos conceptos y crean confusión a la hora de estructurar un proyecto de software.

Palabras claves

Ciclo de vida del software, modelo de ciclo de vida, metodología de desarrollo de software, proceso de desarrollo de software y proyecto de software.

Abstract

General description about the principal process of software development, each one presents advantages and disadvantages.

* Ingeniera de Sistemas de la Universidad Antonio Nariño de Bogotá, Especialista en Multimedia Educativa de la Universidad Antonio Nariño, Especialista en Educación en Tecnología de la Universidad Distrital Francisco José de Caldas, estudiante de Maestría en Ciencias de la Información y las Telecomunicaciones en la Universidad Distrital, docente investigadora del grupo Metis adscrito a la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Correo electrónico: spinzon@udistrital.edu.co.

** Ingeniero de sistemas de la Universidad Central de Bogotá, especialista en Auditoría en Sistemas de Información de la Universidad Católica de Colombia y especialista en Sistemas de Información de la Organización en la Universidad de los Andes, estudiante de Maestría en Ciencias de la Información y las Telecomunicaciones en la Universidad Distrital, coordinador del grupo de Investigación Metis, docente investigador adscrito a la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Correo electrónico: j cguevarab@udistrital.edu.co

News proposals have been appeared and the role of client, the development team and the different management topics has today greater importance. Additionally, they clarify the terms of development software process, models of the software life cycle, methodologies and methods, had mainly to that some authors use indiscriminately without settling down clearly the differences that exist between these concepts and create confusion at the time of structuring a software project.

Key words
Cycle of life of the software, model of cycle of life, methodology of development of software, process of development of software and project of software.

1. Introducción

Desde sus inicios hasta nuestros días, el proceso de desarrollo de software ha sido complejo, debido, en gran medida, a los cambios que se han presentado en las tecnologías de información y las necesidades de las organizaciones para satisfacer los requerimientos de los clientes y el medio que las rodea.

A su vez, el desarrollo de software ha adquirido gran importancia en las organizaciones, puesto que las aplicaciones pueden ser elementos estratégicos y diferenciadores sobre sus competidores y porque implican llevarse a cabo en un tiempo determinado, la utilización de recursos económicos, humanos y físicos limitados, el mantenimiento de las aplicaciones y el cumplimiento de estándares de calidad. En vista de lo anterior, el proceso de desarrollo de productos de software, debe ser organizado y gestionado, a través de proyectos que vayan alineados con las estrategias de la organización.

En un proyecto de software intervienen diversos factores entre los que se destacan las personas, el producto, el proyecto, las herramientas y los procesos que deben ser administrados para llegar a los resultados deseados. Uno de los principales factores de un proyecto de software es el proceso de desarrollo que se va seguir para la implementación de un producto; dentro de este proceso es necesario tener claro las actividades y procesos del ciclo de vida del software, los diferentes modelos del ciclo de vida del software, las metodologías y los métodos para el desarrollo de éste.

El presente artículo muestra de manera general cómo ha evolucionado el software, la forma de gestionar un proyecto de software y sus principales factores; de igual manera, se profundiza en el proceso de desarrollo de éste describiendo el ciclo de vida, algunos de los modelos de ciclo de vida y metodologías de desarrollo de software.

2. Evolución del software

El contexto en el que se han venido desarrollado los proyectos de software está fuertemente ligado a cinco décadas, en las que han evolucionado de los sistemas informáticos [1] y [2]:

1. Década de 1950 a 1960, el hardware sufrió continuos cambios, mientras que el

software se consideraba como un añadido. El software se desarrollaba de manera artesanal. Se utilizaban lenguajes de bajo nivel. El procesamiento se realizaba por lotes. La mayoría del software desarrollado era utilizado por la misma persona u organización. El desarrollo de software carecía de metodologías y se realizaba sin ninguna planifi cación. La documentación no existía y era muy dependiente del hardware.

2. En la década de 1960 a 1970 los sistemas multiusuario introdujeron nuevos con ceptos de interacción hombre-máquina. El procesamiento se realizaba en tiem po real. Los avances en los dispositivos de almacenamiento condujeron a la pri mera generación de sistemas de bases de datos. Se caracterizó por el estableci miento del software como producto y la llegada de las casas de software. Década de lenguajes y compilación. Crisis del software.
3. La década de 1970 a 1980 se caracteriza por la llegada y amplio uso de los com putadores personales. El hardware de los computadores se convierten en un pro ducto estándar. Las ventas de produc tos se incrementaron. El procesamiento distribuido, incrementó la complejidad de los sistemas informáticos. Las redes de área local y global, las comunicacio nes digitales de alto ancho de banda y la creciente demanda de acceso "instantá neo" a los datos, supusieron una fuerte presión sobre los desarrolladores del software. Programación estructurada, In geniería de software y Primeros métodos estructurados.
4. Década de 1980 a 1990: las técnicas de la cuarta generación para el desarrollo del software están cambiando la forma en que la comunidad del software cons truye programas informáticos. Nuevos paradigmas de programación y producción de programas como la orientación a objetos, el cliente servidor, entre otros. Tecnologías de sistemas manej adores de bases de datos y sistemas operativos.
5. Década de 1990 a nuestros días: las metodologías orientadas a objetos están desplazando rápidamente los enfoques de desarrollo de software tradicionales. Análisis orientado a objetos. Diseño orientado a objetos. UML. Tecnología Case de segunda generación. Métodos ágiles. Componentes y reutilización. Interoperabilidad (CORBA, .net, J2EE). Internet. Comercio electrónico. El software libre se está convirtiendo en una tendencia importante.

3. Gestión de proyectos de software

Un proyecto de software es un conjunto de etapas, actividades y tareas necesarias que tienen como objetivo desarrollar un producto de software, dentro de un tiempo, alcance y recursos determinados, los que deben ser gestionados para llegar al resultado propuesto. La división del trabajo en actividades más sencillas permite al personal del proyecto dominar la complejidad del software que se quiere desarrollar.

La gestión de un proyecto de software implica considerar cuatro disciplinas [3] y [4] que actúen sobre su ejecución: planifi - cación, organización, dirección y control, las cuales conforman el ciclo de gestión de proyectos de software y las que se pueden apreciar en la figura 1.

1. Planifi cación: Implica la defi nición de todo el trabajo por realizar; defi nición de los objetivos del proyecto; descomposición de las actividades para asignar res-

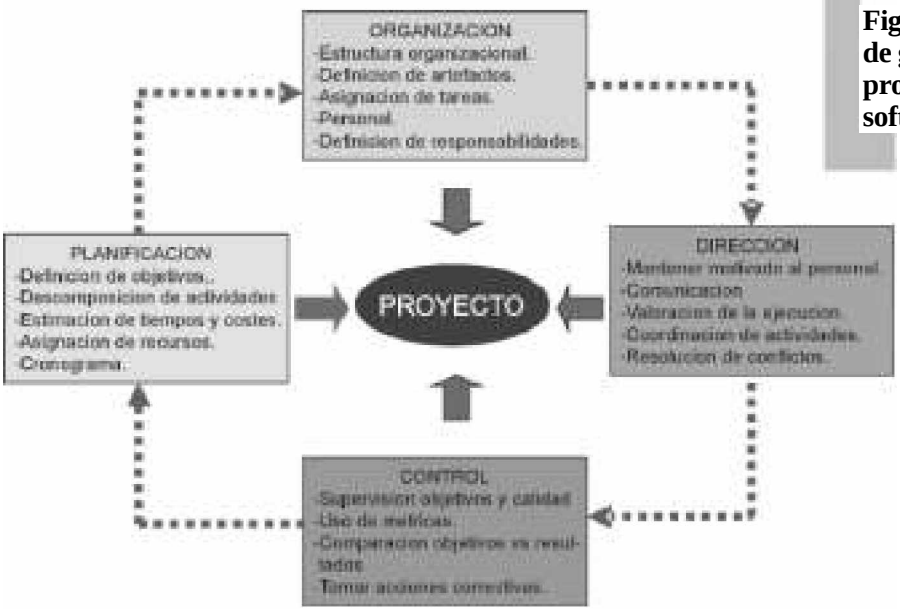


Figura 1. Ciclo de gestión de proyectos de software

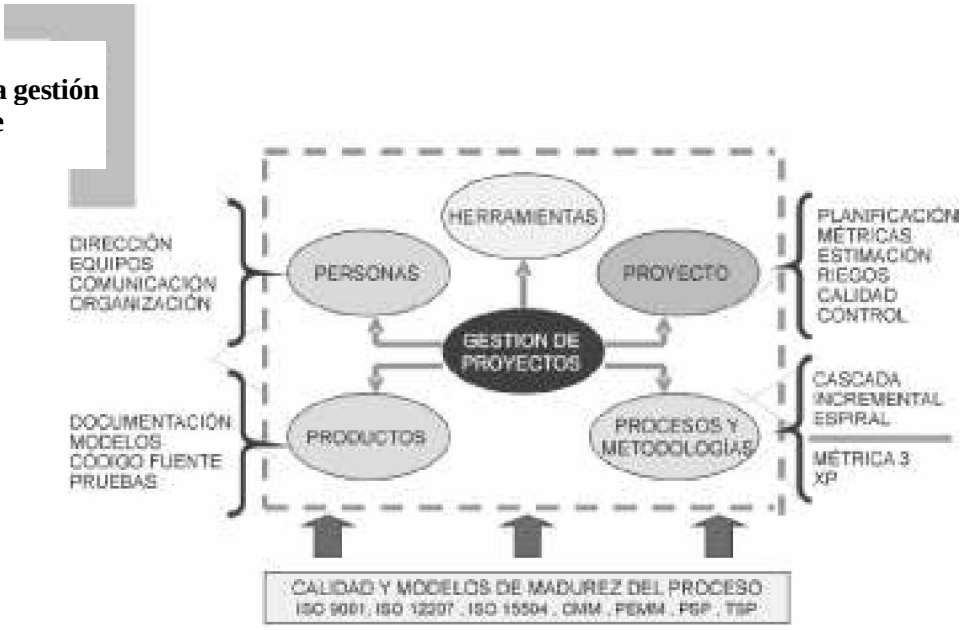
ponsabilidades; relación entre actividades; estimación de tiempos y costes de las actividades; asignación de recursos, cronograma, y obtención y distribución de recursos.

2. Organización: Implica la estructura organizacional del equipo de trabajo; la defi nición de artefactos y responsabilidades que van a utilizar y realizar los integran tes del equipo; descripciones del trabajo; asignación de tareas a puestos de trabajo; personal; determinación de las relaciones organizativas, y delegación de autoridad.
3. Dirección: Implica motivación, comu nicación, jefatura, coordinación, valora ción de la ejecución, resolución de con fl ictos y mantenimiento de las políticas de la empresa.
4. Control: Implica supervisar el cumpli miento de los objetivos y la calidad de los productos, a medida que se desarro lla el proyecto; uso de métricas que permitan establecer el estado del proyecto, en un momento dado; comparación de la ejecución actual y la deseada, y tomar acciones correctivas.

Adicionalmente, en la ejecución de un proyecto de software se deben tener cinco elementos principales: personal, producto, proceso, proyecto y herramientas, los cuales deberán ser administrados a través de las disciplinas para el logro de los objetivos. En la figura 2, se muestran los diferentes elementos de la gestión de un proyecto de software.

1. Personas: Son los autores reales de un proyecto de software, dentro de los que están los ingenieros de requerimientos, analistas, desarrolladores, ingenieros de pruebas, los documentadotes, los investigadores tecnológicos, entre otros, cada uno desempeña un rol diferente. Además de tener bien defi nidos los roles del

Figura 2. Elementos de la gestión de proyectos de software



equipo de trabajo, se debe contar con una estructura organizativa y un buen sistema de comunicación que permita mantener informado al equipo de los compromisos adquiridos y resultados obtenidos, a lo largo del proyecto.

2. Productos: Son el conjunto de artefac tos y resultados que se crean durante la vida del proyecto, como los modelos, el código, los ejecutables, la documenta ción, versiones de productos, entre otros. Antes de poder planifi car un proyecto se deben establecer los objetivos y el ámbito del producto, se debe considerar solucio nes alternativas e identifi car las difi culta des técnicas y de gestión.

3. Proyecto: Es el elemento organizativo a través del que se gestiona el desarrollo del software [5]. El trabajo, a través de proyectos, es la forma habitual de actua ción en el desarrollo de software. En el planteamiento de un proyecto es con- veniente tener en cuenta las siguientes características [6]: un enfoque único, un resultado fi nal específi co, un comienzo y un fi nal, un cronograma para llevarlo a cabo, un trabajo con personas y, de manera interfuncional, recursos limitados, una secuencia de actividades interdependientes y un determinado cliente.

4. Proceso: Es una defi nición del conjunto completo de actividades necesarias para transformar los requisitos de un usuario en un producto. Existen cuatro actividades fundamentales que son comunes para todos los procesos de desarrollo de software [7]: la especifi cación del software, el desarrollo del software, validación del software y la evolución de éste.

5. Herramientas: Son el conjunto de software que se utiliza para automatizar las actividades defi nidas en el proceso. El proceso debe estar soportado por herramientas automatizadas que mejoren la productividad del equipo de desarrollo y la calidad de los productos resultan-

tes, dentro de las que encontramos los CASE.

La calidad de software y los modelos de madures del proceso son dos elementos adicionales que se deben tener en cuenta en el desarrollo de proyectos de software y ayudan a mejorarlo.

El concepto de calidad de software tiene diferentes signifi cados [8], por ejemplo: para la IEEE software [9] es el grado en que un sistema, componente o proceso cumple con los requerimientos especifi cados y las necesidades del cliente o usuario. Para la norma ISO-9000 [10] la calidad del software es el grado en que un conjunto de características inherentes al software cumplen con los requisitos del sistema. La calidad de éste depende, en gran medida, del proceso de desarrollo que se siga.

Los modelos de madurez de la producción de software nos permiten evaluar las capacidades de ingeniería de software de los individuos, equipos y organizaciones y determinar la calidad de los procesos de software. Algunos modelos importantes son PSP, TSP y CMM.

4. Ciclo de vida del software

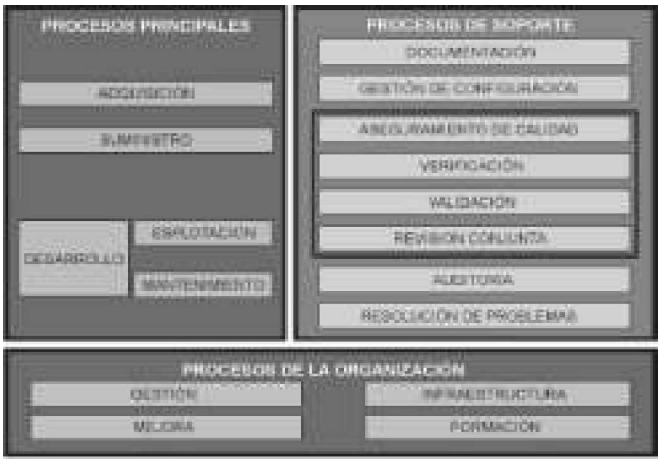
El ciclo de vida del software es una descripción del conjunto de actividades y procesos que permiten dirigir el desarrollo de un producto de software [11].

En la norma ISO 12207-1 [12] las actividades que se pueden realizar, durante el ciclo de vida del software, se agrupan en cinco procesos principales, ocho procesos de soporte y cuatro procesos generales de la organización. En la figura 3, se muestran los procesos del ciclo de vida del software.

Los procesos principales son aquellos que resultan útiles e inician o realizan el desarrollo, la explotación o el mantenimiento del software durante su ciclo de vida; los procesos de soporte son los que sirven de apoyo al resto y se aplican en cualquier punto del ciclo de vida, y los procesos de la organización son los que se emplean para llevar a cabo funciones tales como la gestión, la formación del personal o la mejora del proceso. En la tabla 1, se describen de manera general los procesos principales.

Figura 3.

Procesos del ciclo de vida del software según ISO12207-1



Fuente: Piattini, Mario. *Análisis y diseño detallado de Aplicaciones Informáticas de Gestión.* Alfaomega. 2000, pp.40

Tabla 1. Procesos del ciclo de vida del software

TIPO	PROCESO	DESCRIPCIÓN
Principales	Adquisición	Contiene las actividades y las tareas que el usuario o cliente realiza para adquirir un producto de software.
	Suministro	Contiene las actividades y tareas que un proveedor de un producto de software tiene que realizar para comercializarlo.
	Desarrollo	Contiene las actividades de requerimientos, análisis, diseño, desarrollo, implementación y pruebas.
	Explotación	Contiene las actividades de comercialización del producto de software y las tareas de soporte a usuarios.
	Mantenimiento	Contiene las actividades de modifi cación del producto de software con el objetivo de mantener su consistencia.
Soporte	Documentación	Contiene las actividades que permitan planifi car, diseñar, desarrollar, producir, editar distribuir y mantener documentos necesarios para las personas que utilizan el software.
	Gestión de confi guración	Contiene las actividades para identifi car, defi nir y establecer las características de confi guración del software en un sistema.
	Aseguramiento de calidad	Contiene las actividades que permiten obtener la satisfacción del cliente y el cumplimiento de estándares que la garanticen un buen desarrollo del producto.
	Verifi cación	Contiene las actividades que permiten determinar si los requisitos de un producto de software están completos y correctos.
	Validación	Contiene las actividades que permiten determinar si el producto de software cumple con los requisitos previstos para su uso.
	Revisión conjunta	Contiene las actividades que permitan evaluar el estado del software y sus productos en una actividad del ciclo de vida o una fase de un proyecto.
	Auditoría	Contiene las actividades que permiten determinar, si los puntos establecidos, y si se han cumplido los requisitos, los planes y el contrato.
	Resolución de problemas	Contiene las actividades que permiten analizar y eliminar problemas descubiertos durante el desarrollo, explotación, mantenimiento u otro proceso.
Organización	Gestión	Contiene las actividades y las tareas genéricas que puede emplear una organización que tenga que gestionar sus procesos.
	Infraestructura	Contiene las actividades que permitae establecer y suministrar la infraestructura necesaria para un proceso.
	Mejora	Contiene las actividades para establecer, valorar, medir, controlar y mejorar los procesos del ciclo de vida del software.
	Formación	Contiene las actividades que permiten proporcionar y mantener al personal formado.

5. Modelos del ciclo de vida del software

El modelo de ciclo de vida de software es un marco de trabajo defi nido, que contiene las actividades y tareas que involucradas en el desarrollo, la explotación y mantenimiento de un producto de software y abarca la vida del sistema, desde la defi nición hasta la fi - nalización de su uso [11] y [12].

Existe una gran variedad de modelos entre los que encontramos el de cascada, el incremental y espiral y, dentro de estos, se pueden identifi car componentes como: arquitectura, actividad, métodos y metodologías, estrategia y herramientas para la administración del software [8].

5.1. Modelo del proceso en cascada

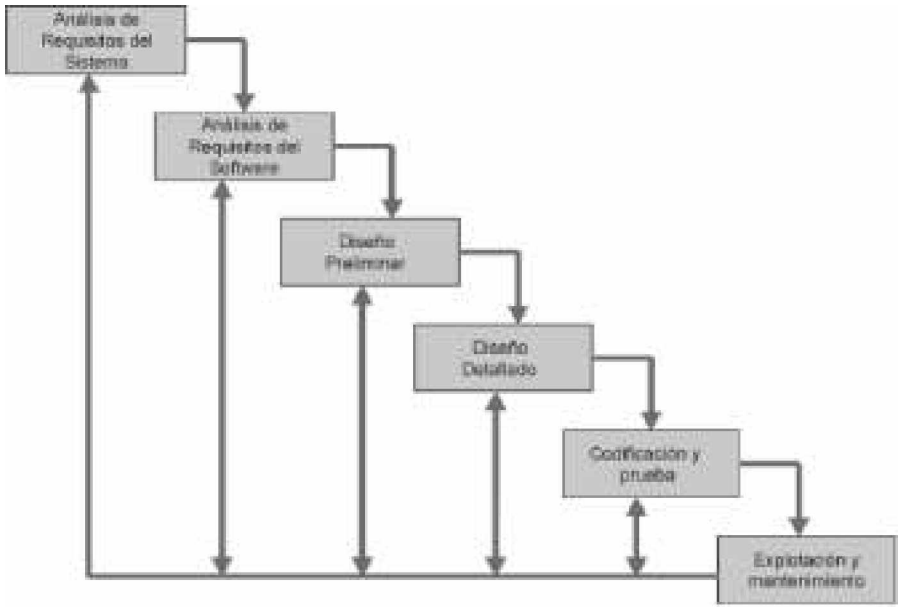
Descripción

La versión original del modelo de cascada del ciclo de vida fue propuesta por Winston Royce [3] y, desde entonces, han aparecido numerosos refi namientos y variaciones de dicho modelo. Este proceso es llamado algunas veces "ciclo de vida básico" o "modelo en cascada" y comprende un enfoque secuencial, para el desarrollo de cada una de las etapas que conforman el proceso (aná-

Tabla 2. Componentes de un modelo de ciclo de vida de software

COMPONENTE DE UN MODELO	DESCRIPCIÓN
Arquitectura	Describe los elementos de la estructura de un programa o sistema (subsistemas, clases, componentes y nodos), sus interrelaciones, y los principios y reglas que gobiernan su diseño y evolución a largo plazo.
Actividad	Describe un paso del proceso de software necesario para lograr los objetivos. Las actividades básicas de un proceso de desarrollo de software son: requisitos, análisis, diseño, implementación, pruebas, documentación y mantenimiento.
Métodos	Describen los procedimientos que defi nen las tareas o acciones que permiten realizar las transformaciones internas de cada actividad.
Metodologías	Describe el conjunto de métodos, técnicas, herramientas y soportes documentales, que defi nen las reglas para realizar las transformaciones internas de las actividades de un modelo de ciclo de vida, que permiten a los desarrolladores implementar nuevo producto de software [14] y [15].
Estrategia	Describe el plan de trabajo y las decisiones que se deben tomar para el logro del objetivo como: la selección de la tecnología, del lenguaje de programación, la especifi cación, entre otras.

Figura 3.
Etapas del modelo del proceso en cascada



Fuente [12]: Piattini, Mario. Análisis y diseño detallado de Aplicaciones Informáticas de Gestión. Alfaomega. 2000.

lisis de requisitos del sistema, análisis de requisitos de software, diseño preliminar, diseño detallado, codifi cación y pruebas y explotación y mantenimiento). En la figura 4, se muestran las etapas del proceso.

Actividades

En el modelo de cascada [8] y [13] se defi ne como una secuencia de actividades que se deben seguir y desarrollar de manera estricta. Las actividades que conforman el modelo son:

1. Análisis de requisitos: establece los requisitos de todos los elementos del sistema y asigna al software algún subgrupo de estos requisitos.
2. Análisis de requisitos de software: determina la naturaleza de los programas por construirse, el dominio de información del mismo, así como la función requerida, comportamiento, rendimiento e interconexión.
3. Diseño preliminar: establece la estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo).
4. El diseño detallado comprende las tareas que permiten realizar una especifi cación detallada de los bloques de construcción, los formatos precisos y el contenido de las salidas del sistema, la especifi cación detallada de los modelos que se van a programar, el diseño de las estructuras de almacenamiento y se especifi can los controles administrativos de entrada a la base de datos y la salida de documentación.

5. Codifi cación: comprende el proceso de llevar los diagramas del diseño detallado al código.
6. Pruebas: comprende las pruebas para la detección de errores y asegurar que la entrada defi nida produce resultados rea les, de acuerdo con los resultados reque ridos.
7. Mantenimiento: cambios que se produ cen cuando se han encontrado errores, cundo el software debe adaptarse para acoplarse a los cambios de su entorno externo.

5.2 Modelo del proceso incremental

Descripción

El modelo incremental permite una secuencia no lineal de los pasos de desarrollo. Aplica secuencias lineales de forma escalonada, mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un "incremento" del software. En el modelo incremental se va creando el sistema de software y se van añadiendo componentes funcionales al sistema. En cada paso sucesivo se actualiza el sistema con nuevas funcionalidades o requisitos, es decir, cada versión o refi namiento parte de una versión previa y le añade nuevas funciones. El sistema de software ya no se ve como una única utilidad monolítica con una fecha fi ja de entrega, sino como una integración de resultados sucesivos obtenidos después de cada iteración, como se muestra en la figura 4.

El modelo incremental se ajusta a entornos de alta incertidumbre, por no tener la necesidad de poseer un conjunto exhaustivo de requisitos, especifi caciones, diseños, etc., al comenzar el sistema, ya que

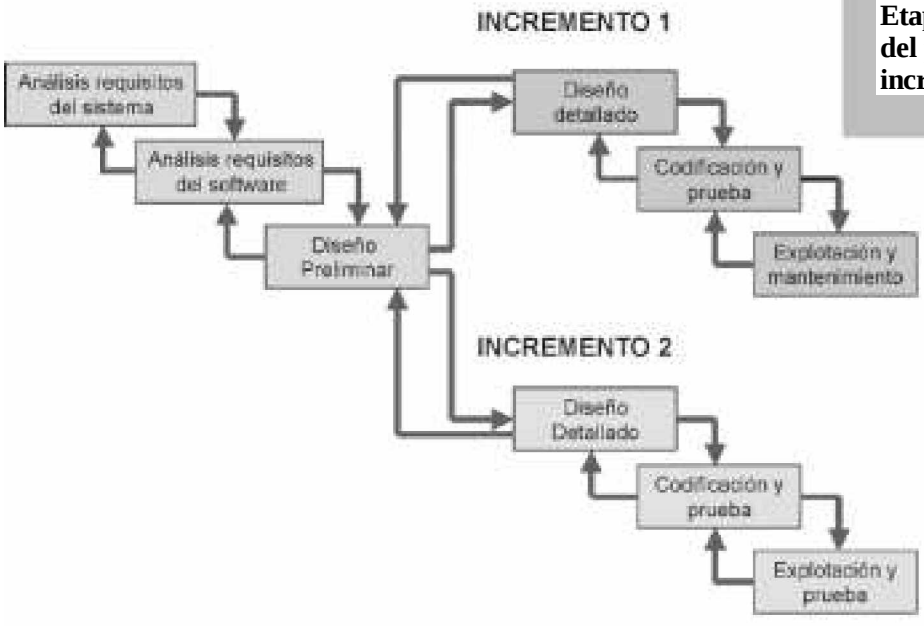


Figura 4. Etapas del modelo del proceso incremental

Fuente [12]: Piattini, *Mario. Análisis y diseño detallado de Aplicaciones Informáticas de Gestión.* Alfaomega. 2000.

cada refi namiento amplía los requisitos y las especifi caciones derivadas de la fase anterior [12]

Actividades

El primer incremento a menudo es un producto esencial, es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer. El cliente utiliza el producto central (o sufre la revisión detallada). Como un resultado de utilización y/o de evaluación se desarrolla un plan para el incremento siguiente. El plan afronta la modifi cación del producto central, con el fi n de cumplir mejor las necesidades del cliente y la entrega de funciones, y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo [13].

5.3. Modelo del proceso en espiral

Descripción

El modelo en espiral del ciclo de vida fue desarrollado por B. Boehm. En éste el esfuerzo de desarrollo es iterativo: tan pronto como se completa una iteración (un módulo o un esfuerzo de desarrollo) se da inicio a la siguiente.

Al igual que todos los modelos evolutivos, el Modelo en Espiral es un modelo iterativo que proporciona en cada iteración una versión evolutiva del producto. En cada iteración se siguen cuatro pasos: determinar qué se desea lograr; determinar las alternativas que pueden ser tomadas, para lograr esas metas (para cada una, analizar los riesgos y resultados fi nales, con el fi n de seleccionar la mejor), y seguir la alternativa selecciona-

Figura 5.
Modelo de ciclo de vida en espiral de Bohem

Fuente [18] disponible en:http://weblogs.udp.cl/masterti/archivos/ (2363)RiesgosProcesos Metodos AgilesdeSoftware.ppt

da y establecer qué se ha completado, hasta dónde se ha llegado y qué esta faltando.

Durante las primeras etapas, la versión incremental podría ser un prototipo o un modelo en papel. Durante las últimas iteraciones, se producen versiones cada vez más completas del software.

Actividades

Este modelo se divide en regiones de tareas. Generalmente, existen entre tres y seis. En la figura 5 se ve un modelo en espiral que contiene seis regiones de tareas: Comunicación con el Cliente; Planifi cación (se defi nen recursos y tiempo); Análisis de Riesgos (se evalúan riesgos técnicos y de gestión); Ingeniería (se construyen modelos de la aplicación a desarrollar); Construcción y Entrega (se construye, prueba, instala y proporciona soporte al usuario); Evaluación del Cliente.

El proceso comienza desde el centro, girando en el sentido de las agujas del reloj. El primer circuito en gris más oscuro alrededor del espiral (múltiples iteraciones pueden ocurrir dentro de un circuito) podría resultar en el desarrollo de una especifi cación del producto; sucesivas pasadas podrían ser usadas para desarrollar un prototipo y progresivamente versiones más sofi sticadas del software.

A diferencia del modelo lineal secuencial que termina cuando se entrega el software, el modelo en espiral puede ser adaptado para ser aplicado a un proyecto que se encuentre en cualquier punto del ciclo de desarrollo. Empieza con una recolección de requerimientos y una planifi cación inicial, luego se realiza un análisis de riesgo basado en los requerimientos iniciales. Posteriormente, se determina si es conveniente seguir o no a partir de los riesgos del proyecto. Después se obtiene un prototipo inicial de software, que es presentado al cliente para ser evaluado. Este proceso se repite hasta que la evaluación del cliente sea positiva y se hayan cumplido con los requerimientos establecidos.

5.4. Ventajas y desventajas de los modelos

Los modelos de cascada, incremental y espiral presentan ventajas y desventajas que deben ser tenidas en cuenta al seleccionarlos para el desarrollo de un aplicativo.

Tabla 3. Ventajas y desventajas de los modelos de desarrollo de software

VENTAJAS Y DESVENTAJAS	CASCADA	INCREMENTAL	ESPIRAL
Ventajas	**1.** Proporciona una planilla, en la que se encuentran métodos para análisis, diseño, codifi cación, pruebas y mantenimiento.	1. La administración de proyectos es más fácil de lograr en incrementos más pequeños.	1. Combina la naturaleza interactiva del modelo de prototipos, con los aspectos controlados y sistemáticos del proceso lineal.

VENTAJAS Y DESVEN-TAJAS	CASCADA	INCREMENTAL	ESPIRAL
Ventajas	2. Es el proceso de desarrollo de software más antiguo y utilizado para la implementación de aplicaciones.	2. Es más fácil comprender y comprobar incrementos de funcionalidad más pequeños.	2. Introduce la planifi - cación y el análisis de riesgo, como elementos para tener en cuenta, en el desarrollo de proyectos de software.
Desventajas	1. El proceso de desarrollo de software, generalmente, es imposible llevarlo a cabo de esta manera, ya que los proyectos reales raramente siguen este fl ujo secuencial, puesto que siempre hay iteraciones. 2. A menudo es difícil que el cliente exponga explícitamente todos los requisitos. El modelo lineal secuencial lo requiere y tiene difi cultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos.	1. El proceso de desarrollo de software su puede tornar interminable sino se defi nen claramente los alcances del proyecto al comenzarlo. 2. Aunque permite el cambio continuo de requisitos, aún existe el problema de determinar si los requisitos propuestos son válidos. Los errores en los requisitos se detectan tarde y su corrección resulta tan costosa como en el modelo de cascada.	1. Requiere mucha habilidad para el análisis de riesgos y de esta habilidad depende su éxito. 2. No ha sido utilizado tanto como el lineal secuencial o el de prototipos

6. Metodologías

Una metodología es un conjunto de métodos, procedimientos, técnicas, herramientas y soportes documentales que defi nen las reglas para realizar las transformaciones internas de las actividades de un modelo de ciclo de vida, que permiten a los desarrolladores implementar nuevo producto de software [14] y [15].

Una metodología está conformada por un conjunto de componentes que especifi can: cómo se debe dividir el proyecto en etapas, qué tareas se llevan a cabo en cada etapa, qué salidas se producen y cuándo se deben producir, qué restricciones se aplican, qué herramientas se van a utilizar y cómo se gestiona y controla un proyecto.

Una metodología puede seguir uno o varios modelos de ciclos de vida, esto es: el ciclo de vida indica qué es lo que hay que obtener, a lo largo del desarrollo del proyecto, pero no cómo. Esto sí lo debe indicar la metodología. Los enfoques metodológicos han ido evolucionando a través del tiempo, en los que se destacan las metodologías estructuradas, las orientadas a objetos y las ágiles En la tabla 4, se describen algunas generalidades de las metodologías estructuradas o tradicionales y las orientadas a objetos.

En la tabla 5 se muestran algunas de las principales metodologías estructuras y orientadas a objetos [12].

Tabla 4. Descripción de las metodologías estructuradas y orientadas a objetos

	METODOLOGÍAS ESTRUCTURADAS	METODOLOGÍAS ORIENTADAS A OBJETOS
Enfoque	Principalmente, en la descomposición funcional de un sistema. El objetivo es lograr una descomposición completa en términos de funciones, estableciendo los datos de entrada y salida correspondientes.	Principalmente, en el modelo de un sistema en términos de objetos. A diferencia de las metodologías estructuradas, se identifi can inicialmente los objetos del sistema, para luego especifi car su comportamiento
Herramientas de modelado	1. Diagramas de fl ujos de datos, que sirven para modelar la transformación de datos entre funciones del sistema. 2. Diagramas de transición de estados que sirven para modelar el comportamiento en el tiempo. Describen el efecto de eventos externos en los eventos y funciones. 3. Diagramas de entidad-relación, sirven para modelar el almacenamiento de los datos.	1. Diagramas de clase: Sirven para describir los componentes esenciales de la arquitectura de un sistema. 2. Diagramas de casos de uso: Especifi can un sistema en término de su funcionalidad. 3. Diagramas de transición de estado: Describen los cambios de estado en los objetos, siendo equivalentes sus similares en las metodologías estructuradas. 4. Diagramas de secuencia: Sirven para describir los aspectos dinámicos del sistema, mostrando el fl ujo de eventos entre objetos en el tiempo. 5. Diagramas de colaboración: se utilizan para describir la comunicación entre objetos de un sistema. 6. Diagramas de subsistemas: se usan para describir agrupaciones de clases en un sistema.

Tabla 5. Metodologías estructuras y orientadas a objetos

TIPO	METODOLOGÍA	FECHA	TIPO	METODOLOGÍA	FECHA
Estructurada	Merise	1978	Orientada a objetos	SYNTHESIS	1989
	SSADM	1981		OOSD	1990
	SSADM Versión 3	1986		OMT	1991
	Métrica Versión inicial	1989		Fusión	1994
	SSADM Versión 4	1990		OOA/D	1994
	Métrica Versión 2	1993		MOSES	1994
	Métrica Versión 2.1	1995		Syntropy	1994
	Métrica Versión 3	2001		MEDEA	1994
				RUP	1998

6.1 Metodología PUD o RUP

El Proceso Unifi cado de Desarrollo de Software (PUD) fue creado por Jacobson, Booch y Rumbaugh. Este proceso se deriva de metodologías anteriores desarrolladas por estos tres autores, a saber, la metodología Objectory de Jacobson, la metodología de Booch y la técnica de modelado de objetos de Rumbaugh et al. [16].

El PUD, como anotan sus autores:

> Es un proceso de desarrollo de software basado en el Lenguaje Unifi cado de Modelado (UML), y que es iterativo, centrado en la arquitectura y dirigido por los casos de uso y los riesgos. Proceso que se organiza en cuatro fases: inicio, elaboración, construcción y transición, y que se estructura en torno a cinco fl ujos de trabajo fundamentales: recopilación de requisitos, análisis, diseño, implementación y pruebas [16].

Características

Las características principales del PUD o RUP con las que se describen a continuación [15]:

Etapas

El PUD es un proceso evolutivo y se desarrolla a través de una serie de ciclos que constituyen la vida de un sistema y se concluyen con una versión del producto, desde su inicio hasta su muerte. Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición. Las fases se dividen en un conjunto de iteraciones, en las que se desarrollan los fl ujos de trabajo: requerimientos, análisis, diseño, implementación y pruebas.

Tabla 6. Características del RUP

CARACTERÍSTICAS	DESCIPCIÓN
Está dirigido por casos de uso	Un proyecto progresa a través de las iteraciones, en las que se llevan a cabo los fl ujos de trabajo, los cuales inician a partir de los casos de uso. Un caso de uso es una descripción de las acciones de un sistema, desde el punto de vista del usuario. Los casos de uso ayudan a los desarrolladores a encontrar las clases e implementar las interfaces.
Es centrado en la arquitectura	Un proyecto se centra en la arquitectura, es decir, en la defi - nición de decisiones signifi cativas acerca de la organización del sistema de software, la defi nición de los elementos estructurales (subsistemas, clases, componentes y nodos), de los que se compone y las interfaces entre ellos, junto con su comportamiento, que permitan la construcción del sistema, garantizando un progreso continuo, no sólo para la versión en curso, sino también para la vida entera del producto.
Es iterativo e incremental	Un proyecto se construye a través de las iteraciones, que permiten generar los modelos resultantes incremento por incremento. Cada iteración añade algo más a cada modelo, a medida que la iteración fl uye a lo largo de requisitos, análisis, diseño, implementación y prueba.

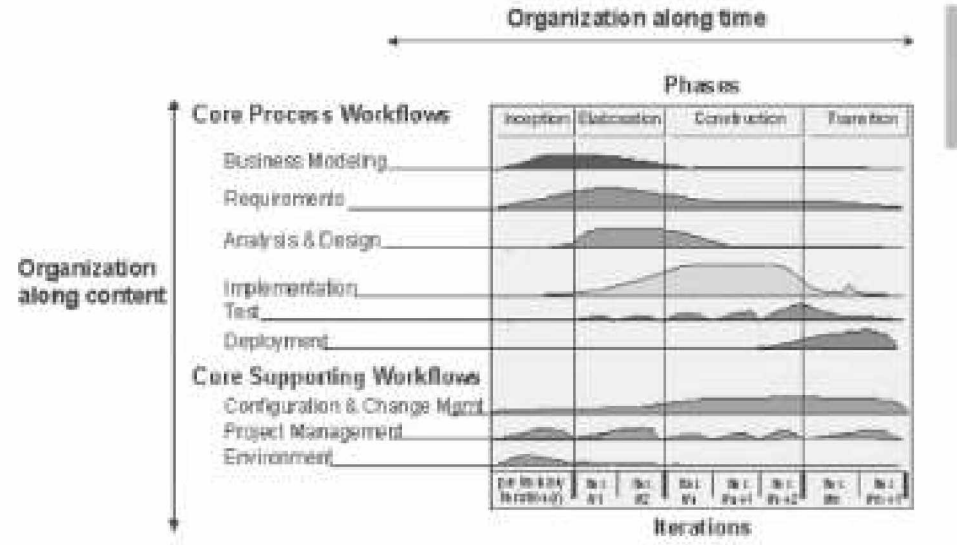


Figura 6.
Proceso de Desarrollo Unifi cado

Fuente [17] disponible en: http://www.dcc.uchile.cl/~luguerre/cc61j/recursos/clase2.ppt

Las fases y las principales actividades que se desarrollan en cada una de ellas se describen a continuación en la tabla 7.

Por su parte los fl ujos de trabajo y cada una de sus respectivas actividades que se realizan dentro de las fasesse encuentran en la tabla 8 [19]:

6.2. Metodología Métrica Versión 3

Métrica Versión 3 fue desarrollada por el Ministerio de Administraciones Públicas de España, está orientada al desarrollo de sistemas de información y da soporte al ciclo de vida de sistemas en una organización,

Tabla 7.
Fases de un ciclo

FASES	ACTIVIDADES
Inicio	Identifi cación de los requerimientos generales del proyecto. Identifi cación y especifi cación de todos los casos de uso. Identifi cación de actores. Identifi cación de recursos necesarios tanto económicos como humanos, de acuerdo al proyecto. Realizar una estimación detallada de tiempos y recursos.
Elaboración	Defi nición de la arquitectura de acuerdo con los principales casos de uso Análisis del dominio del problema o negocio. Determinar la viabilidad del proyecto y decidir si se continúa con él. Escribir los planes de trabajo de las etapas de construcción y transición.
Construcción	Completar la funcionalidad del sistema, clarifi cando los requerimientos pendientes. Administrar el cambio de artefactos construidos. Ejecutar plan de administración de recursos.
Transición	Asegurar la disponibilidad del software para los usuarios fi nales. Hacer un proceso de detección y corrección de errores. Capacitación de usuarios y provisión de soporte técnico. Verifi car concordancia entre el producto fi nal y los requerimientos de los usuarios.

Tabla 8.
Flujo de trabajo

FLUJO DE TRABAJO	DESCRIPCIÓN	ACTIVIDADES
Modelamiento del negocio	Se realiza una descripción de los procesos del negocio y se establece una visión general del sistema.	Describir los procesos del negocio. Defi nir el modelo del dominio. Establecer el glosario de términos.
Requerimientos	Se desarrolla un modelo del sistema que se va construir a través de los casos de uso, que permiten defi nir la funcionalidad (operaciones) que debe cumplir el sistema.	Defi nir actores. Determinar lista preliminar de casos de uso. Depurar casos de uso. Modelo de casos de uso. Documentar.
Análisis	Se estudian los requisitos, refi nándolos y estructurándolos, con el objeto de conseguir una comprensión más precisa que nos permita estructurar el sistema entero en un modelo de objetos conceptual.	Diagramas de secuencia Diagramas de colaboración. Diagramas de actividad. Diagramas de estado. Modelo de análisis.
Diseño	Se describe la manera de implementar físicamente los casos de uso, encontrando la arquitectura necesaria que permita soportarlos, incluyendo los requisitos no funcionales y otras restricciones.	Lista inicial de objetos. Defi nir responsabilidades. Defi nir colaboraciones. Modelo de diseño. Modelo objeto relacional. Diccionario de datos. Diagrama de despliegue. Descripción de la arquitectura. Implementación.
Pruebas		Modelo de pruebas. Pruebas individuales. Pruebas del sistema.

para garantizar la calidad y efi ciencia de los productos desarrollados. Esta metodología se basa en las versiones anteriores (Métrica versión inicial, métrica versión 2 y versión 2.1) e incorpora métodos de desarrollo más extendidos, así como los últimos estándares de ingeniería del software y calidad y referencias específi cas referentes a seguridad y gestión de proyectos. Además, se han tenido en cuenta las opiniones de los usuarios, para mejorar algunos problemas y defi ciencias detectados [20].

Por otra parte, Métrica Versión 3, cubre dos tipos de desarrollo: estructurado y orientado a objetos, de esta forma, el proyecto puede incorporar diferentes puntos de vista y facilitar el proceso de desarrollo.

Conclusiones

Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no

cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y fi nales.

En cuanto a la versatilidad de los modelos de proceso podemos decir que existen varias corrientes; unas se catalogan como modelos de proceso ligero, con practicas simples, productivas y realistas como XP y otros modelos de proceso de grano grueso, que buscan controlar, en forma fuerte, las variables, etapas, roles y artefactos en forma disciplinada y más consistente. Las denominadas metodologías ágiles han aparecido como respuesta a la complejidad aportada en procesos como PUD y Métrica 3, pero distan de ofrecer un marco genérico de conducción de procesos que sea aplicable a proyectos de software de mediana o alta complejidad.

Referencias bibliográfi cas

[1] Pressman, Roger. (1993). *Ingeniería de Software. Un enfoque práctico.* s.d.: Editorial Mc Graw Hill. Tercera Edición, pp. 4 -7.

[2] Ros, Joaquín Nicolás. (s.d.). *Tema 1. Introducción a la ingeniería de software.* Universidad de Murcia. Disponible en: http://dis.um.es/~jnicolas/09BK_FIS.html

[3] Royce, Walter. (1998). *Software Project Management.* s.d.: Editorial Addison Wesley, pp. 139.

[4] Piattini, Mario G., Calvo, José A., Cervera, Joaquín, Fernández, Luis. (2004). *Análisis y diseño detallado de Aplicaciones Informáticas de Gestión.* s.d.: Editorial Alfaomega, pp. 90.

[5] Jacobson, Ivar, Booch, Grady, Rumbaugh, James. (1999). *El Proceso Unifi cado de Desarrollo de Software*. s.d.: Editorial Addison Wesley, pp. 13, 431.

[6] Randolph, Alan, Posner, Barry. (1993). *Gerencia de proyectos. Cómo dirigir exitosamente equipos de trabajo.* s.d.: Editorial Mc Graw Hill, p. 3.

[7] Sommerville, Ian. (2002). *Ingeniería de Software.* s.d.: Editorial Addison Wesley Sexta Edición, p. 43.

[8] Weitzenfeld, Alfredo. (2005). *Ingeniería de Software. Orientada a objetos con UML, Java e Internet.* s.d.: Editorial Thomson, pp. 50, 56.

[9] ANSI / IEEE Std. 729 (1983). *IEEE Standard Glossary of Software Engineering Terminology.* Nueva York: IEEE Inc.

[10] ISO 9000-3. Disponible en: http://www.iso.org

[11] (1997). "Software Engineering Standards Comite of the IEEE Computer Society". *IEEE Standard for Developing Software Life Cycle.* IEEE Std. 1074, pp. 13.

[12] Piattini, Mario G., Calvo, José A., Cervera, Joaquín, Fernández, Luis. (2000). *Análisis y diseño detallado de Aplicaciones Informáticas de Gestión.* s.d.: Editorial Alfaomega, pp. 39, 40.

[13] Bernd, Bruegge, Dutoit. (2002). *Ingeniería de software orientada a objetos.* s.d.: Editorial Prentice Hall, p. 13.

[14] Rumbaugh, James, Blaha, Michael, Premerlani, William, Hedí Frederic, Lorensen, Wiliam. (1999). *Modelado y diseño orientado a objetos Metodología OMT.* s.d.: Editorial Prentice Hall, p. 197.

[15] Braude, Eric. (2003). *Ingeniería de Software. Una perspectiva orientada a objetos*. Alfaomega, p. 29.

[16] Guerrero, Luis A. (s.d.). *Racional Unifi ed Process. CC61J Taller de UML*. Departamento de Ciencias de la Computación. Universidad de Chile. Disponible en: http://www.dcc.uchile.cl/~luguerre/cc61j/recursos/clase2.ppt

[17] Visconti, Marcello. (s.d.). *Riesgos, procesos y métodos ágiles de software.* Departamento de informática. Universidad Técnica Federico Santa María. Disponible en: http://weblogs.udp.cl/masterti/archivos/