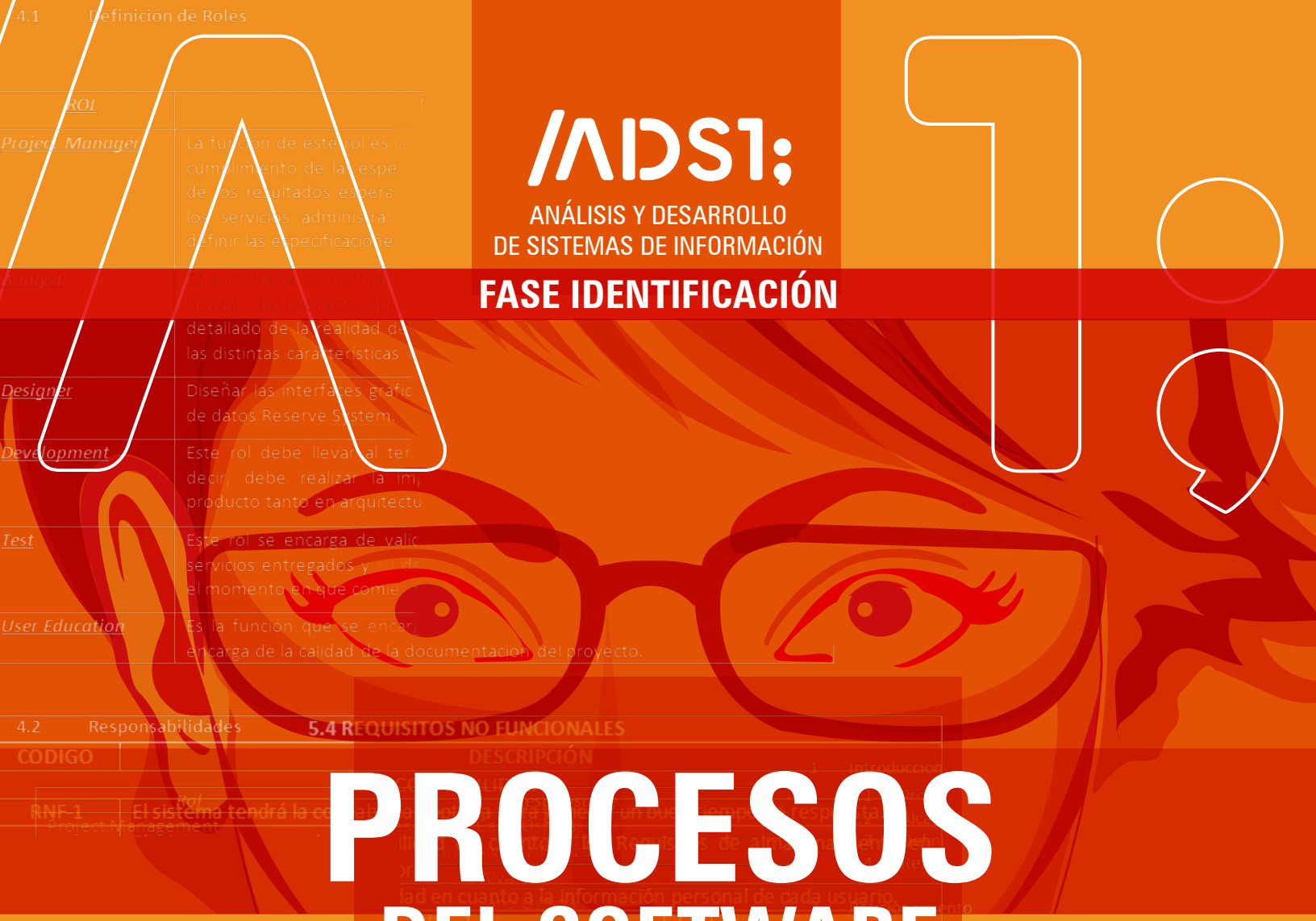
ADSI:

ANÁLISIS Y DESARROLLO
DE SISTEMAS DE INFORMACIÓN

FASE IDENTIFICACIÓN



PROCESOS DEL SOFTWARE

PROCESO DEL SOFTWARE - MODELOS
PROCESOS DEL SOFTWARE COMERCIALES O METODOLOGÍAS DE DESARROLLO

ACTIVIDAD DE PROYECTO

1. Determinar las especificaciones funcionales del Sistema de Información.

ACTIVIDAD DE APRENDIZAJE

2. Diseñar los mapas de procesos de las áreas involucradas en el sistema de información a desarrollar.

SENA

De clase mundial

PROCESO DEL SOFTWARE

El software se construye de la misma forma que cualquier otro producto: aplicando un proceso de producción, dicho de otra manera siguiendo una serie predecible de pasos, por tanto involucra modelos de calidad de proceso y de producto. Un proceso del software está formado por un conjunto de actividades y resultados asociados que generan un producto de software.

La existencia de un proceso de software no es garantía que el software será entregado a tiempo, que satisfaga al cliente y cumpla con sus expectativas. Para el aseguramiento de la calidad en los procesos del software se trabaja a la par estándares de calidad. Los ISO 9000 y 9001 son normas genéricas que actualmente se aplican a cualquier organización que desee conseguir el aseguramiento de la calidad de sus productos, sistemas o servicios que provee.

Aunque existen muchos procesos diferentes de software algunas actividades fundamentales son comunes para todos ellos como lo plantea (*Sommerville*, 2005):

1. **Especificación de software o ingeniería de requerimientos:** debe definir la funcionalidad del software y las restricciones en su operación.
2. **Diseño e implementación del software,** se debe producir software que cumple su especificación.
3. **Validación del software,** se debe validar el software para asegurar que hace lo que el cliente desea.
4. **Evolución del software,** el software debe evolucionar para cubrir las necesidades cambiantes del cliente.

Otros autores anteriores plantean el ciclo de vida del desarrollo de software con las etapas de Análisis, Diseño, Desarrollo (o Codificación), Implementación, Pruebas y Evolución. Etapas que concuerdan con las etapas genéricas de un proceso de software planteadas por (*Sommerville*, 2005).

La conclusión es que todos los procesos del software involucran un ciclo de vida, así como el ciclo de vida del ser humano es nacer, crecer, reproducirse y morir; los software poseen también su **ciclo de vida.** *"Ciclo de vida del SW es el marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto SW, abarcando desde la definición de los requisitos hasta la finalización de su uso".*

MODELOS DEL PROCESO DEL SOFTWARE

Un modelo de proceso del software es una descripción simplificada de un proceso del software, es decir es el modelo o “molde, base” del cual se basa el proceso del software. El modelo genérico o modelo básico se denomina Ciclo de Vida del Software.

Los modelos planteados en este numeral son denominados modelos descriptivos es decir cualquier organización constructora de software puede describir un conjunto único de actividades dentro del marco de trabajo para el (los) proceso(s) de software que adopte. Sin importar el modelo del proceso seleccionado, el equipo de desarrollo debe elegir de manera tradicional un marco de trabajo genérico para el proceso, el cual puede incluir las etapas propuestas anteriormente, seguir el ciclo de vida o definir un marco basado en: comunicación, planeación, modelado, construcción y desarrollo. Note que siguen siendo las mismas etapas solo que varían su nombre.

Modelo de cascada

Es el más conocido, está basado en el ciclo de vida tradicional del software, el paradigma del ciclo de vida abarca las siguientes actividades:

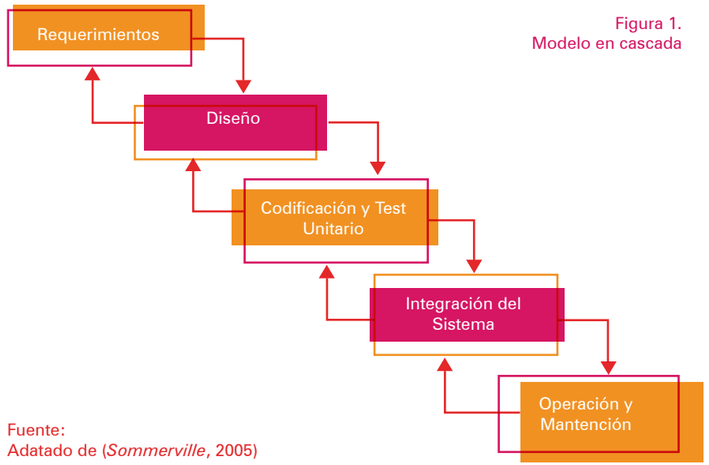


Figura 1.
Modelo en cascada

Fuente:
Adatado de (*Sommerville*, 2005)

Análisis de los requisitos del software: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

Diseño: el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

Codificación: el diseño debe traducirse en una forma legible para la maquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

Integración y pruebas: una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Operación y mantenimiento: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a que hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Desventajas:

- Los proyectos reales raramente siguen el flujo secuencial que propone el modelo, siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
- Normalmente, es difícil para el cliente establecer explícitamente al principio todos los requisitos. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.
- El cliente debe tener paciencia. Hasta llegar a las etapas finales del proyecto, no estará disponible una versión operativa del programa. Un error importante no detectado hasta que el programa esté funcionando puede ser desastroso.

La ventaja de este método radica en su sencillez ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software

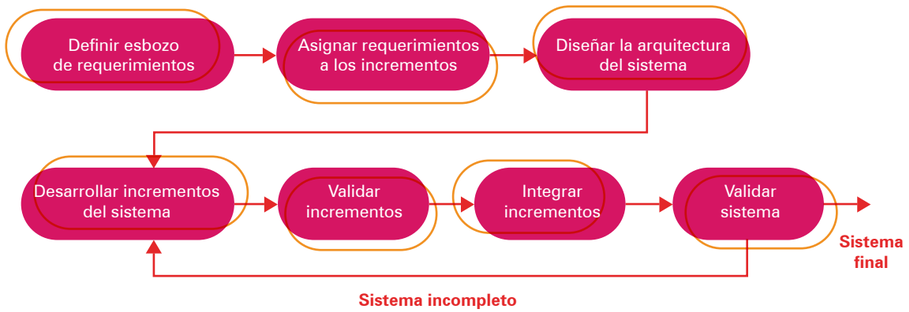
MODELOS DE PROCESO INCREMENTALES

En muchas situaciones los requisitos iniciales del software están bien definidos en forma razonable, pero el enfoque global del esfuerzo de desarrollo excluye un proceso puramente lineal. Además, quizá haya una necesidad imperiosa de proporcionar de manera rápida un conjunto limitado de funcionalidad para el usuario y después refinarla y expandirla en las entregas posteriores del software. En estos casos se elige un modelo de proceso diseñado para producir el software en forma incremental (Pressman, 2005).

Modelo Incremental

La figura 2 presenta el modelo de proceso incremental. En un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos. Entonces, se derivan varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio con los servicios de prioridad más alta entregados primero (*Sommerville*, 2005).

Figura 2.
Modelo de proceso incremental



Fuente:
(*Sommerville*, 2005)

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos paralos incrementos posteriores y para las últimas versiones del incremento actual. Tan pronto como se completan los nuevos incrementos, se integran en los existentes de tal forma que la funcionalidad del sistema mejora con cada incremento entregado. Los servicios comunes se pueden implementar al inicio del proceso o de forma incremental tan pronto como sean requeridos por un incremento.

En (Sommerville, 2005) se plantean algunas ventajas de este modelo:

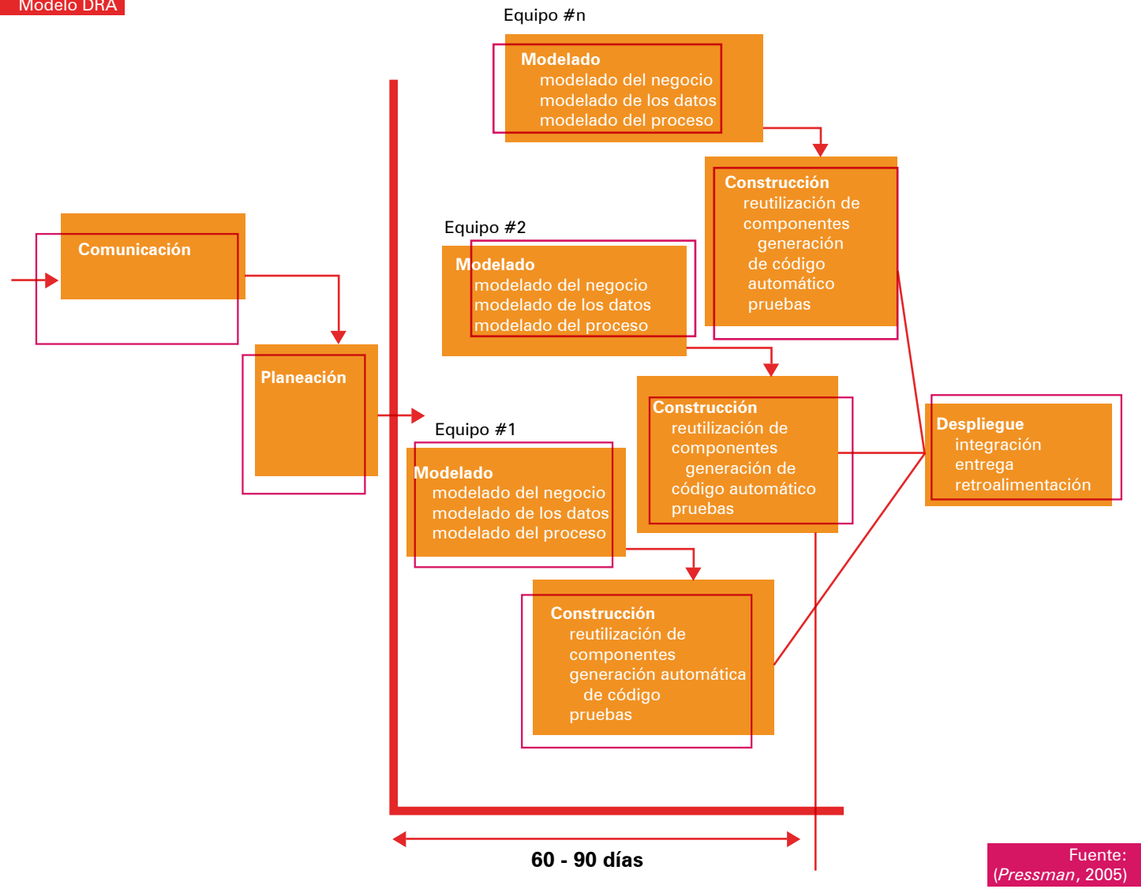
- Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de tal forma que pueden utilizar el software inmediatamente.
- Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.
- Existe un bajo riesgo de un fallo total del proyecto. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue de forma satisfactoria al cliente.
- Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

Sin embargo, existen algunos problemas en el desarrollo incremental. Los incrementos deben ser relativamente pequeños (no más de 20.000 líneas de código) y cada uno debe entregar alguna funcionalidad del sistema. Puede ser difícil adaptar los requerimientos del cliente a incrementos de tamaño apropiado. Más aún, muchos de los sistemas requieren un conjunto de recursos que se utilizan en diferentes partes del sistema. Puesto que los requerimientos no se definen en detalle hasta que un incremento se implementa, puede ser difícil identificar los recursos comunes que requieren todos los incrementos.

Desarrollo Rápido de Aplicaciones, DRA

Modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. Es una adaptación a "Alta velocidad" en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de periodos cortos de tiempo (*Pressman*, 2005). La figura 3 presenta el modelo DRA.

Figura 3.
Modelo DRA



Fuente:
(*Pressman*, 2005)

Fase identificación

Ventajas de este modelo:

- Permite el desarrollo de productos en periodos cortos de tiempo.
- Permite integrar diferentes recursos humanos.
- Enfatiza el desarrollo de componentes de programas reutilizables. La reutilización es la piedra angular de las tecnologías de objetos, y se encuentra en el modelo de proceso de ensamblaje.

En (Pressman, 2005) se establecen algunas desventajas para este modelo:

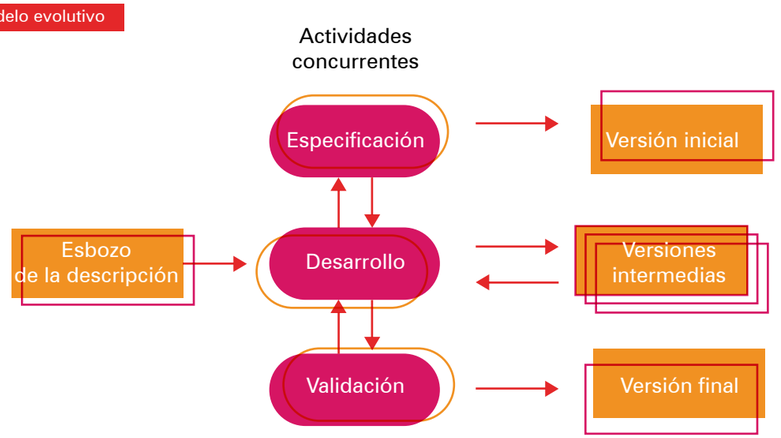
- Para proyectos grandes aunque por escalas, el DRA requiere recursos humanos suficientes como para crear el número correcto de equipos DRA.
- Requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso, por ninguna de las partes constituyentes, los proyectos DRA fracasarán.
- No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede dividir en módulos adecuadamente, la construcción de los componentes necesarios para DRA presentará problemas.
- No es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el nuevo software requiere un alto grado de interoperabilidad con programas de computadora ya existentes.

MODELOS DE PROCESOS EVOLUTIVOS

El software, como todos los sistemas complejos, evoluciona con el tiempo. Los modelos de proceso evolutivos de se basan en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

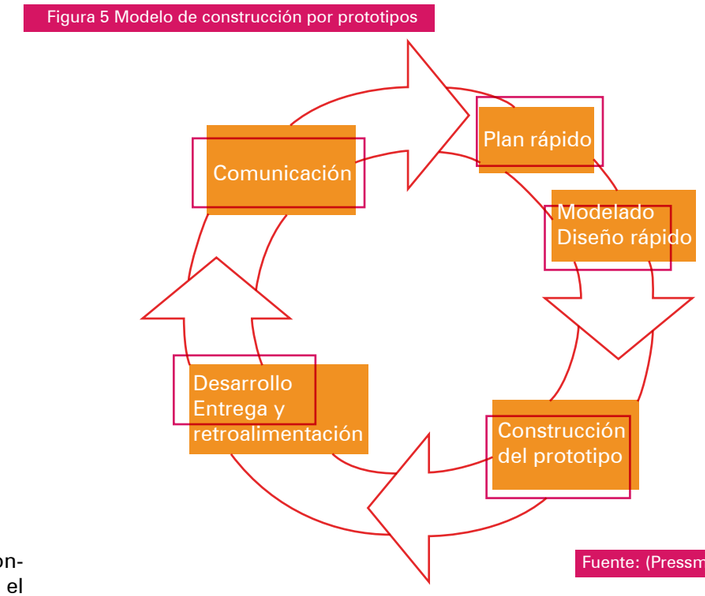
Las actividades de especificación, desarrollo y validación se entrelazan, en vez de separarse, con una rápida retroalimentación entre éstas como se muestra en la figura 4

Figura 4. Modelo evolutivo



Fuente: (*Sommerville,* 2005)

Figura 5 Modelo de construcción por prototipos



Fuente: (Pressman, 2005)

Construcción de prototipos

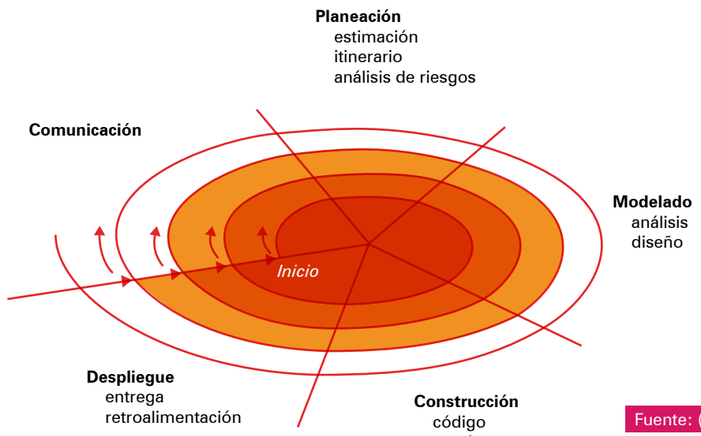
A menudo un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. En otros casos, el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina. En estas, y muchas otras situaciones, un paradigma de construcción de prototipos puede ofrecer la mejor alternativa para el desarrollo (*Pressman*, 2005).

La figura 5 ilustra que la construcción de prototipos se inicia con la comunicación. El analista y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es necesaria más definición. Entonces se plantea con rapidez una iteración de construcción de prototipos y se presenta el modelado (en la forma de un diseño rápido). El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final (por ejemplo, la configuración de la interfaz con el usuario y el formato de los despliegues de salida). El diseño rápido conduce a la construcción de un prototipo. Después, el prototipo lo evalúa el cliente/usuario y con la retroalimentación se refinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente (*Pressman*, 2005)

Modelo en espiral

Propuesto originalmente por *Boehm*, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Cuando se aplica el modelo en espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, la entrega tal vez sea un documento del modelo o un prototipo. Durante las últimas iteraciones se producen versiones cada vez más completas del sistema desarrollado. Un proceso en espiral se divide en un conjunto de actividades del marco de trabajo que define el equipo de ingeniería del software. La figura 6 plantea una serie de etapas que como se indicó deben ser definidas por el equipo de desarrollo. (*Pressman*, 2005)

Figura 6.
Modelo en espiral



Fuente: (*Pressman,* 2005)

Cada una de las actividades del marco de trabajo representa un segmento de la ruta en espiral que se presenta en la figura. Cuando comienza este proceso evolutivo el equipo de software realiza actividades implicadas en un circuito alrededor de la espiral que tiene una dirección en el sentido del movimiento de las manecillas del reloj, y que se inicia desde el centro. El riesgo es un factor considerado en cada revolución. El primer circuito alrededor de la espiral quizá genere el desarrollo de una especificación del producto; los pasos subsecuentes alrededor de la espiral se pueden aprovechar para desarrollar un prototipo y después, en forma progresiva, versiones más elaboradas del software. Cada paso a través de la región de planeación resulta en ajustes al plan del proyecto. Los costos y el itinerario se ajustan con base en la retroalimentación derivada de la relación con el cliente después de la entrega. Además, el administrador del proyecto ajusta el número de iteraciones planeado que se requiere para completar el software (*Pressman*, 2005).

Modelos Orientados a la Reutilización

En la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, esto pasa cuando las personas que trabajan en el proyecto conocen diseños o código similares al requerido. Los buscan, los modifican acorde a lo requerido y los incorporan en el sistema. La reutilización es a menudo indispensable para el desarrollo rápido de sistemas. Esta reutilización informal es independiente del proceso genérico que se utilice. Sin embargo, en años recientes, ha surgido un enfoque de desarrollo de software (ingeniería de software basada en componentes) que se basa en la reutilización, el cual se está utilizando de forma amplia.

El modelo orientado a la reutilización tiene la ventaja obvia de reducir la cantidad de software a desarrollarse y también reduce los costos y los riesgos. Por lo general, también conduce a una entrega más rápida del software. Sin embargo, los compromisos en los requerimientos son inevitables y esto conduce a un sistema que no cumple las necesidades reales de los usuarios. Más aún, si las nuevas versiones de los componentes reutilizables no están bajo la vigilancia de la organización que los utiliza, se pierde el control sobre la evolución del sistema (*Sommerville*, 2005).

CASES

PROCESOS DEL SOFTWARE COMERCIALES O METODOLOGÍAS DE DESARROLLO

Actualmente la decisión de cual proceso de producción de software emplear para desarrollo está determinado por dos tendencias, los procesos tradicionales o pesados y los procesos ágiles:

- **Procesos tradicionales**: procesos donde lo importante es la documentación detallada de cada parte del desarrollo del producto software, usualmente estos procesos involucran aseguramiento de la calidad. La interacción con el cliente es planificada y acordada previamente. Denominados también procesos pesados o no ágiles. Ejemplos de estos procesos se tienen: Proceso Unificado de Desarrollo (RUP), Métrica 3, MSF entre otros.

COMERCIA

- **Procesos ágiles**: procesos cuya característica es la gran interacción que tienen los analistas con todos los implicados en el desarrollo. La documentación que se realiza de los componentes del software se reduce a lo más mínimo pasando a un segundo plano. Ejemplos de estos procesos: XP, SCRUM, FDD entre otros.

GLOSARIO

Arquitectura de software: [Bass et al., 1998]: La arquitectura de un programa o sistema de computación es la estructura o estructuras del sistema, que están compuestas de componentes software, de las propiedades visibles de esos componentes, y las relaciones entre ellos.

Ciclo de vida del software: El conjunto de procesos sistemáticos que tienen lugar durante la existencia del producto, desde su concepción inicial hasta que la organización decide no continuar manteniéndolo.

Extreme Programming (XP): Metodología heterodoxa de programación. Es la más popular de las denominadas metodologías ágiles. Surgida a partir de la metodología de trabajo empleada Kent Beck, Wark Cunningham y Martin Fowler en el desarrollo del proyecto C3 para Chrysler. Extreme Programming (XP) se funda en cuatro valores: comunicación, simplicidad, feedback y coraje.

Feature Driven Development (FDD): Metodología ágil de desarrollo. No requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso. FDD consiste en cinco procesos secuenciales durante los que se diseña y construye el sistema: Desarrollo del modelo general – Construcción de la lista de rasgos – Planeamiento por rasgo – Diseño por rasgo – Construcción por rasgo.

Metodologías ágiles: Estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas; centradas en las personas o los equipos, iterativas, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa de cliente.

Microsoft Solutions Framework (MSF): Marco para desarrollo de sistemas de software basado en principios, modelos, disciplinas, conceptos, prácticas y recomendaciones propias, derivadas de la experiencia de Microsoft. Se autodefine como "marco" y no como metodología, porque considera que no hay una única estructura de procesos válida para todos los proyectos. El marco MSF se adapta de forma flexible a las características de cada proyecto.

OO (Orientación por Objetos): Enfoque para el desarrollo de sistemas de software que representa el dominio de aplicación de forma natural y directa basándose en los objetos que se implican en dicho dominio.

Emplea diversos métodos para representar de forma abstracta los objetos, definiendo su estructura, comportamiento, agrupaciones, estados, etc.

Las estrategias de orientación por objetos han desarrollado metodologías tanto para requisitos, como para análisis, diseño y programación.

Rational Unified Process (RUP): Proceso de Ingeniería del Software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades en las organizaciones de desarrollo de software. Se trata de un proceso integrado en un producto, desarrollado y mantenido por Racional Software, e integrado en su conjunto de herramientas de desarrollo. Se encuentra disponible a través de IBM.

SCRUM: Metodología ágil, aplicada originalmente por Jeff Sutherland y elaborada más formalmente por Ken Schwaber. Scrum aplica principios de control industrial, junto con experiencias metodológicas de Microsoft, Borland y Hewlet Packard.

•Debrauwer, L., & Heyde, F. (2009). *UML* (segunda edición ed.). España, Barcelona: eni ediciones

•Pressman, R. (2005). *Ingeniería del software*, *un enfoque práctico* (sexta edición ed.). McGraw-Hill.

REFERENCIAS

•Sommerville, I. (2005). *Ingeniería del software* (sexta edición ed.). madrid: Pearson Educación.

Glosario de Ingeniería del Software. Recuperado de:

http://www.planetacodigo.com/wiki/glosario:indice el 10 de Agosto de 2012.

LÍDER DEL PROGRAMA ADSI
Vanessa Cristina Miranda Cano
vanessa24@misena.edu.co

ASESORÍA PEDAGÓGICA
Claudia Herrera Cifuentes
pipelore@yahoo.com

ILUSTRACIÓN PORTADA
Saúl Suaza
ssuaza@gmail.com

COMPILACIÓN Y PREPARACIÓN
Jorge Antonio Blanco Velandia

LÍDER LÍNEA DE PRODUCCIÓN
Iliana Eneth Molina Cuarta
ilmocu@sena.edu.co

DIAGRAMACIÓN
Ricardo Burbano Martínez
ribuma@gmail.com

DISEÑO EDITORIAL Y PORTADA
Ricardo Burbano Martínez
ribuma@gmail.com