EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Especialidad Análisis, diseño y desarrollo de software

Análisis Avanzados de Software

D0.2 Modelos de ciclo de desarrollo de software tradicional y ágil

Asesor: M.T.I.C. Leonardo Enriquez
Ingeniero electrónico, sistemas digitales

Contenido

- Modelos de proceso de software tradicionales
  - Prescriptivos: Cascada, incremental, evolutivo, concurrentes.
  - Especializados: Orientado a la reutilización o de componentes, Unificado, Personal o de equipo.
- Modelos de proceso de software ágil
  - Desarrollo dinámico
  - Programación extrema
  - Scrum
- SweBok 3.0 (10 Áreas del conocimiento)
  - Requisitos, Diseño, Desarrollo, Pruebas, Mantenimiento, Gestión de configuración, gestión de software, proceso de ingeniería, herramientas y métodos de ingeniería, calidad del software

Modelos de proceso de software tradicionales

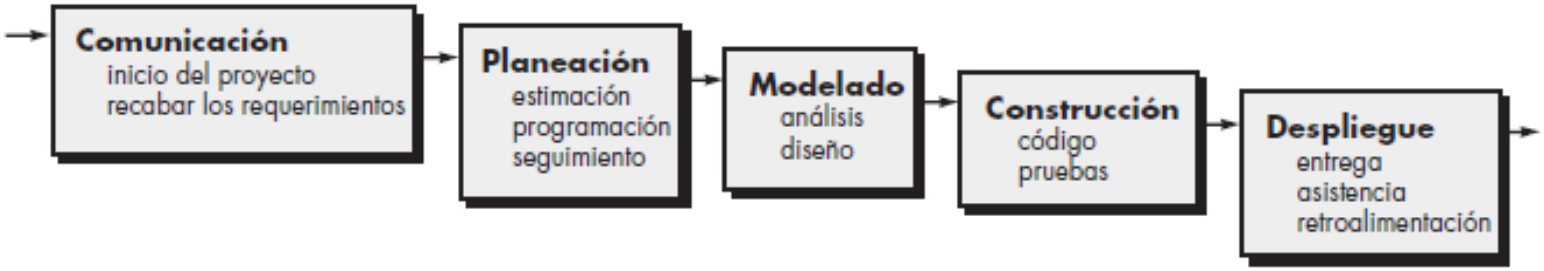
Un proceso de software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software. Estas actividades pueden incluir el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. Sin embargo, las aplicaciones de negocios no se desarrollan precisamente de esta forma. El nuevo software empresarial con frecuencia ahora se desarrolla extendiendo y modificando los sistemas existentes, o configurando e integrando el software comercial o componentes del sistema.

Tipos de Modelos	Tipos de proceso de desarrollo
1. Prescriptivo	• Modelo en Cascada • Modelo de desarrollo incremental • Modelo de proceso evolutivo • Modelo concurrentes
2. De proceso especializado	• Basado en componentes • Proceso unificado • Proceso personal del software PPS • Proceso del equipo de software (PES)

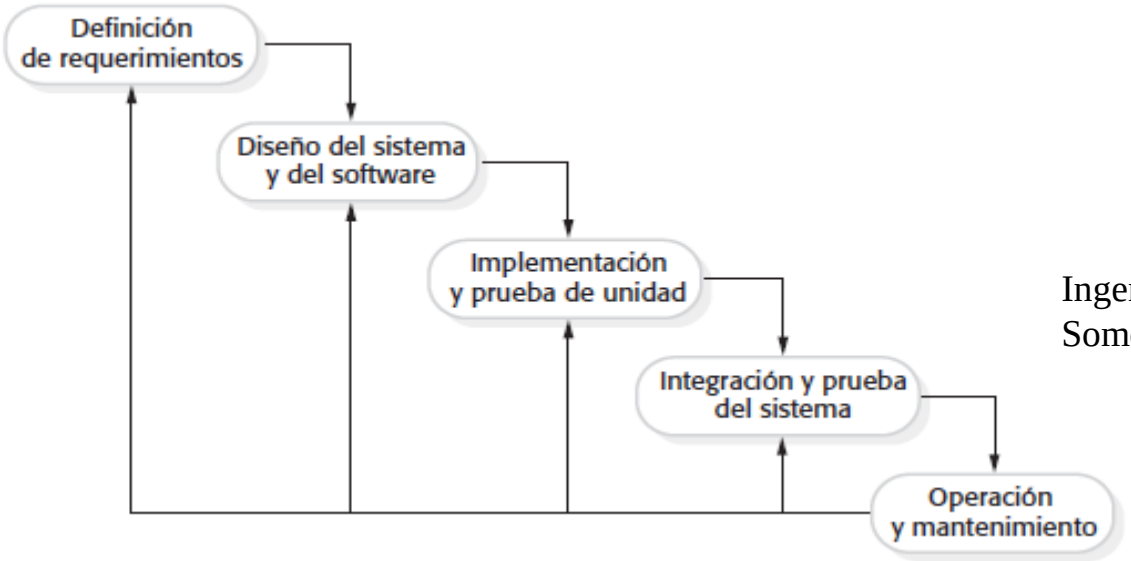
Los modelos de proceso prescriptivo fueron propuestos originalmente para poner orden en el caos del desarrollo de software.

Modelos de proceso prescriptivo: Cascada

El *modelo de la cascada*, a veces llamado *ciclo de vida clásico*, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado



Ingeniería de software Enfoque practico por Pressman 7Ed



Ingeniería de software por Somerville 9Ed

Modelos de proceso prescriptivo: Cascada

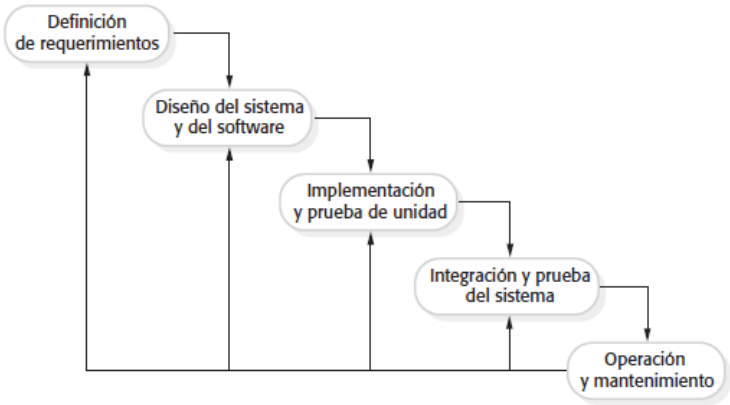
1. ***Análisis y definición de requerimientos*** Los servicios, las restricciones y las metas del sistema se establecen mediante consulta a los usuarios del sistema. Luego, se definen con detalle y sirven como una especificación del sistema.

2. ***Diseño del sistema y del software*** El proceso de diseño de sistemas asigna los requerimientos, para sistemas de hardware o de software, al establecer una arquitectura de sistema global. El diseño del software implica identificar y describir las abstracciones fundamentales del sistema de software y sus relaciones.

3. ***Implementación y prueba de unidad*** Durante esta etapa, el diseño de software se realiza como un conjunto de programas o unidades del programa. La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.

4. ***Integración y prueba de sistema*** Las unidades del programa o los programas individuales se integran y prueban como un sistema completo para asegurarse de que se cumplan los requerimientos de software. Después de probarlo, se libera el sistema de software al cliente.

5. ***Operación y mantenimiento*** Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida, donde el sistema se instala y se pone en práctica. El mantenimiento incluye corregir los errores que no se detectaron en etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e incrementar los servicios del sistema conforme se descubren nuevos requerimientos.



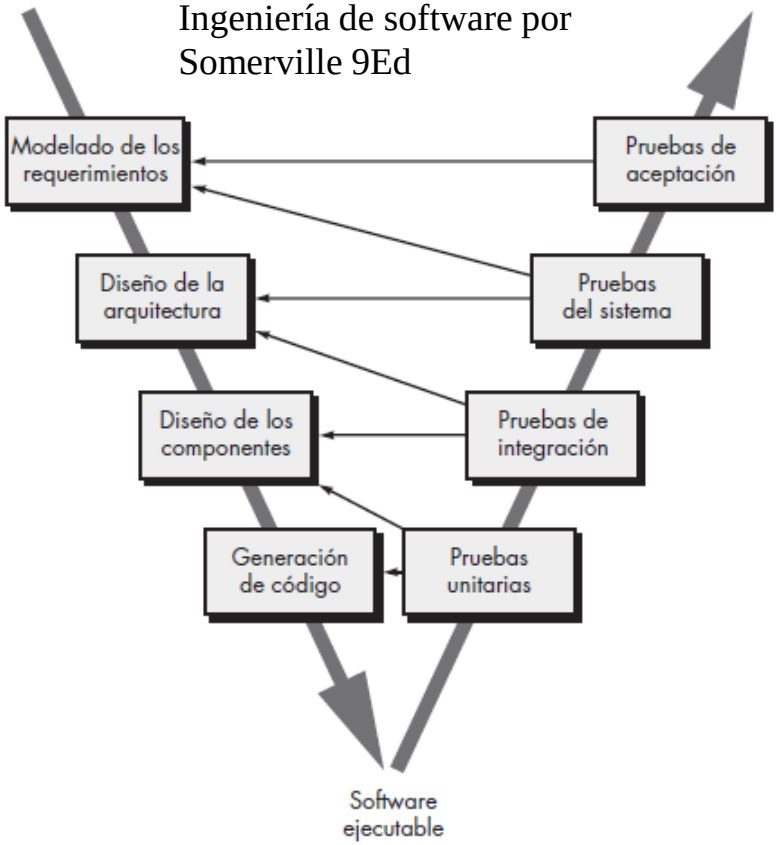
Ingeniería de software por Somerville 9Ed

Modelos de proceso prescriptivo: Variante del Cascada

El modelo de la cascada es el paradigma más antiguo de la ingeniería de software. Sin embargo, en las últimas tres décadas, las críticas hechas al modelo han ocasionado que incluso sus defensores más obstinados cuestionen su eficacia [Han95]. Entre los problemas que en ocasiones surgen al aplicar el modelo de la cascada se encuentran los siguientes:

1. Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo. Aunque el modelo lineal acepta repeticiones, lo hace en forma indirecta. Como resultado, los cambios generan confusión conforme el equipo del proyecto avanza.
2. A menudo, es **difícil para el cliente enunciar** en forma explícita todos los **requerimientos**. El modelo de la cascada necesita que se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.
3. **El cliente debe tener paciencia**. No se dispondrá de una versión funcional del (de los) programa(s) hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento.

Ingeniería de software por Somerville 9Ed



Modelos de proceso prescriptivo: Desarrollo incremental

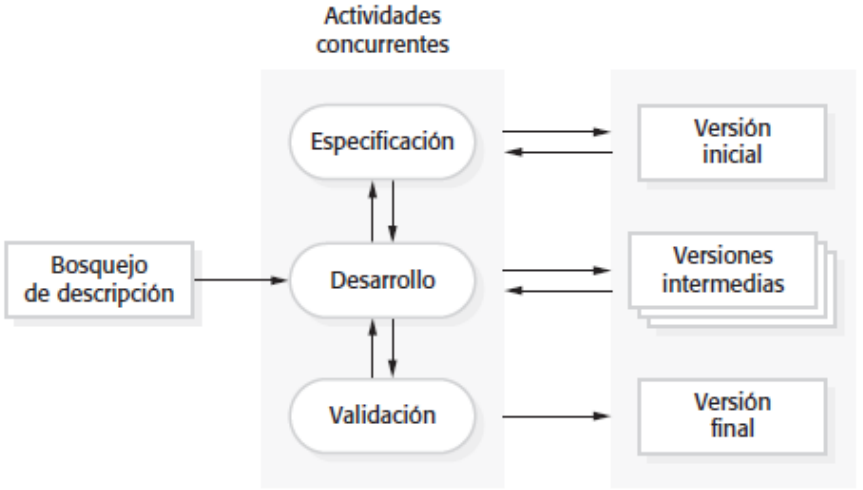
El desarrollo incremental se basa en la idea de diseñar una implementación inicial, exponer ésta al comentario del usuario, y luego desarrollarla en sus diversas versiones hasta producir un sistema adecuado.

Comparado con el modelo en cascada, el desarrollo incremental tiene tres beneficios importantes:

1. Se **reduce** el **costo** de adaptar los requerimientos cambiantes del cliente. La cantidad de análisis y la documentación que tiene que reelaborarse son mucho menores de lo requerido con el modelo en cascada.

2. Es más **sencillo** obtener **retroalimentación** del cliente sobre el trabajo de desarrollo que se realizó. Los clientes pueden comentar las demostraciones del software y darse cuenta de cuánto se ha implementado. Los clientes encuentran difícil juzgar el avance a partir de documentos de diseño de software.

3. Es posible que sea más **rápida** la **entrega** e implementación de software útil al cliente, aun si no se ha incluido toda la funcionalidad. Los clientes tienen posibilidad de usar y ganar valor del software más temprano de lo que sería posible con un proceso en cascada

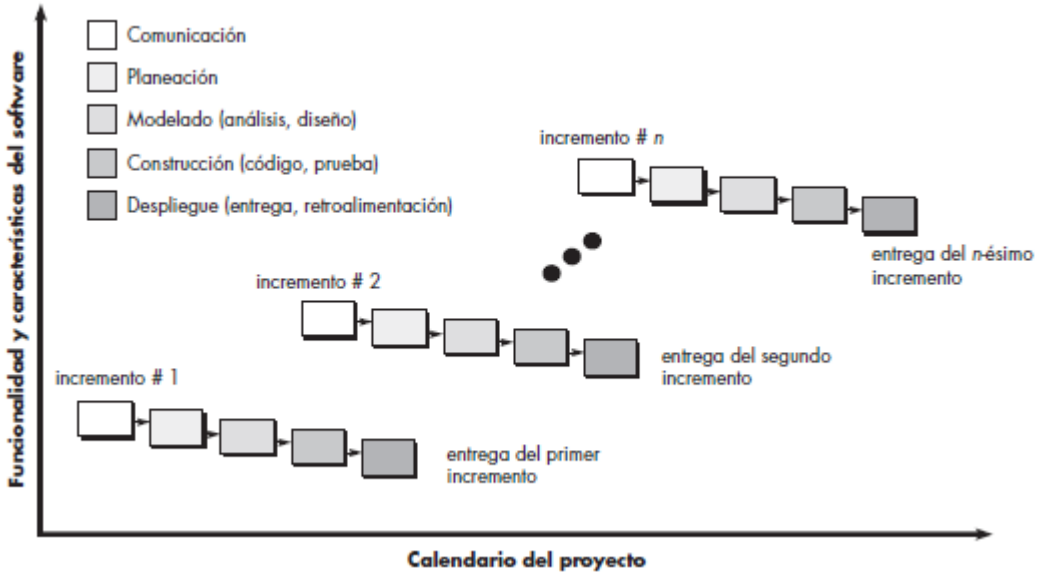


Ingeniería de software por Somerville 9Ed

Modelos de proceso prescriptivo: Desarrollo incremental

El modelo de proceso incremental se centra en que en cada incremento se entrega un producto que ya opera. Los primeros incrementos son versiones desnudas del producto final, pero proporcionan capacidad que sirve al usuario y también le dan una plataforma de evaluación.

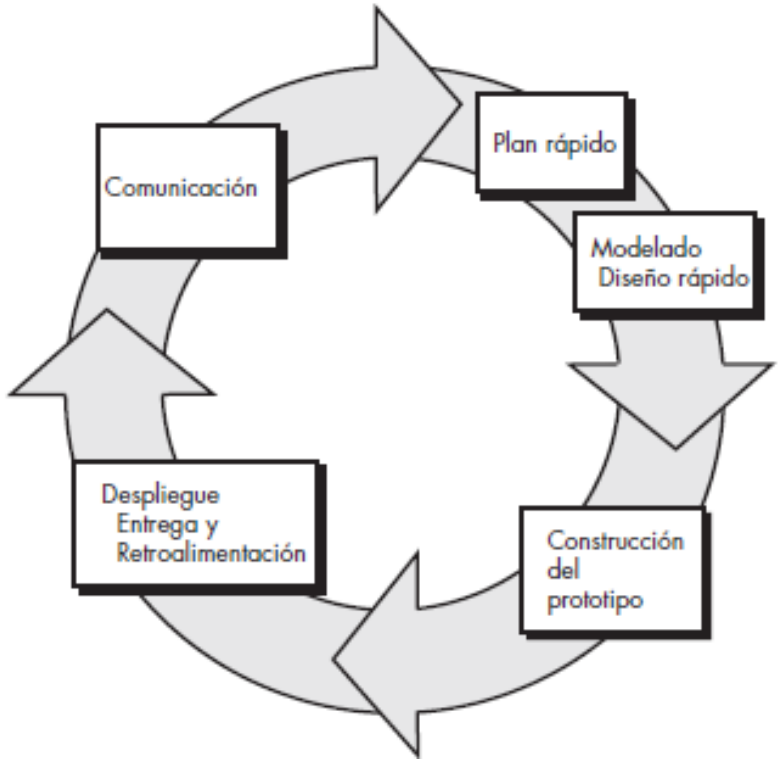
- El desarrollo incremental es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio.
- Los primeros incrementos se desarrollan con pocos trabajadores.
- Si el producto básico es bien recibido, entonces se agrega más personal (si se requiere) para que labore en el siguiente incremento.



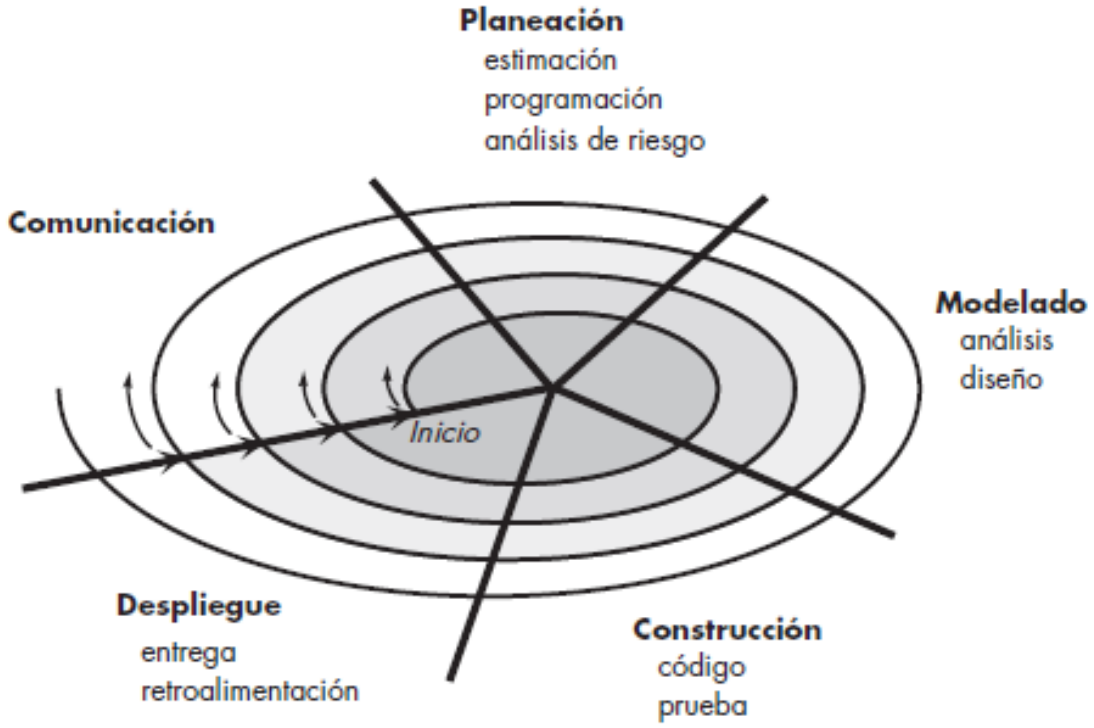
Ingeniería de software Enfoque practico por Pressman 7Ed

Modelos de proceso prescriptivo: Proceso evolutivo Prototipos y Espiral

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software.



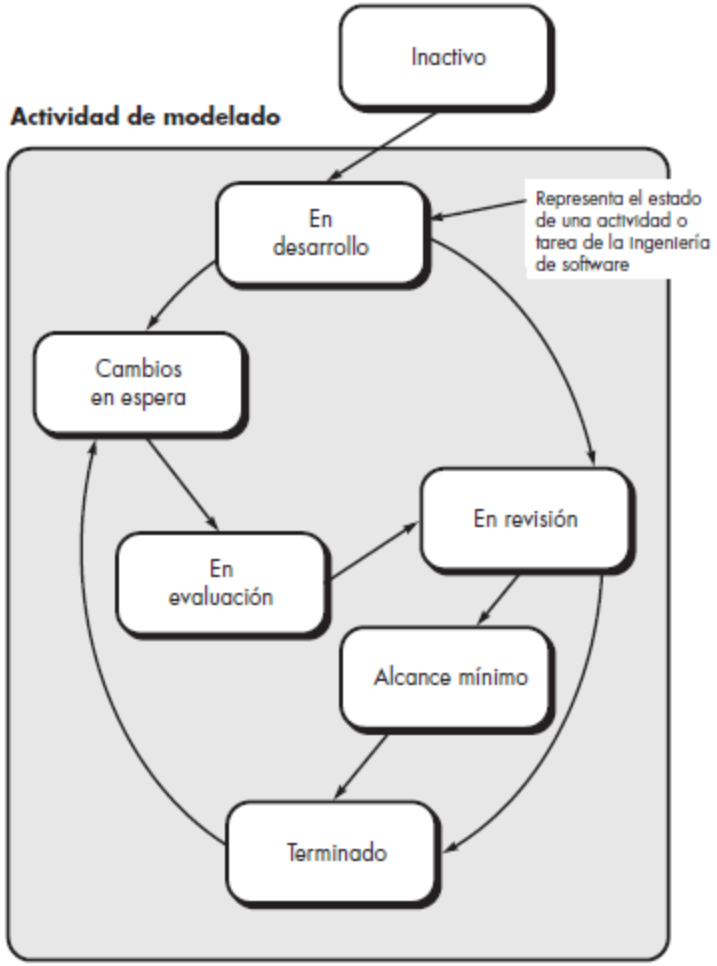
Modelo evolutivo de Prototipos



Modelo evolutivo Espiral

Modelos de proceso prescriptivo: Proceso concurrente

El *modelo de desarrollo concurrente*, en ocasiones llamado *ingeniería concurrente*, permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso descritos en este capítulo. Por ejemplo, la actividad de modelado definida para el modelo espiral se logra por medio de invocar una o más de las siguientes acciones de software: hacer prototipos, análisis y diseño



Modelos de proceso especializado: Orientado a la reutilización

En la mayoría de los proyectos de software hay cierta reutilización de software. Sucede con frecuencia de manera informal, cuando las personas que trabajan en el proyecto conocen diseños o códigos que son similares a lo que se requiere. Los buscan, los modifican según se necesite y los incorporan en sus sistemas.

Etapas	Descripción
1. Análisis de componentes	Dada la especificación de requerimientos, se realiza una búsqueda de componentes para implementar dicha especificación. Por lo general, no hay coincidencia exacta y los componentes que se usan proporcionan sólo parte de la funcionalidad requerida
2. Modificación de requerimientos	Durante esta etapa se analizan los requerimientos usando información de los componentes descubiertos. Luego se modifican para reflejar los componentes disponibles. Donde las modificaciones son imposibles, puede regresarse a la actividad de análisis de componentes para buscar soluciones alternativas.
3. Diseño de sistema con reutilización	Durante esta fase se diseña el marco conceptual del sistema o se reutiliza un marco conceptual existente. Los creadores toman en cuenta los componentes que se reutilizan y organizan el marco de referencia para atenderlo. Es posible que deba diseñarse algo de software nuevo, si no están disponibles los componentes reutilizables.
4. Desarrollo e integración	Se diseña el software que no puede procurarse de manera externa, y se integran los componentes y los sistemas COTS o Sistemas comerciales (off-the-shelf), para crear el nuevo sistema. La integración del sistema, en este modelo, puede ser parte del proceso de desarrollo, en vez de una actividad independiente

Modelos de proceso especializado: Orientado a la reutilización

Existen tres tipos de componentes de software que pueden usarse en un proceso orientado a la reutilización:

Tipos de componentes de software
1. **Servicios Web** que se desarrollan en concordancia para atender servicios estándares y que están disponibles para la invocación remota.
2. **Colecciones de objetos** que se desarrollan como un paquete para su integración con un marco de componentes como .NET o J2EE.
3. **Sistemas de software independientes** que se configuran para usar en un entorno particular.

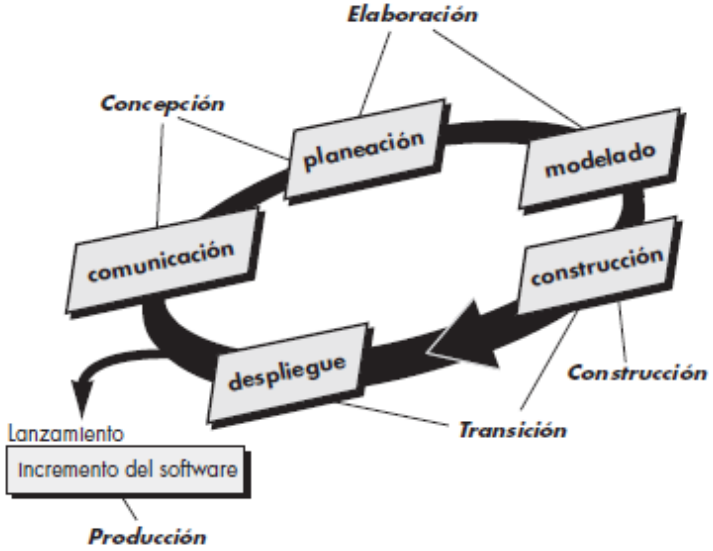
Modelos de proceso especializado: Proceso Unificado Racional RUP

El RUP reconoce que los modelos de proceso convencionales presentan una sola visión del proceso. En contraste, el RUP por lo general se describe desde tres perspectivas:

1. Una perspectiva **dinámica** que muestra las fases del modelo a través del tiempo.

2. Una perspectiva **estática** que presenta las actividades del proceso que se establecen.

3. Una perspectiva **práctica** que sugiere buenas prácticas a usar durante el proceso.

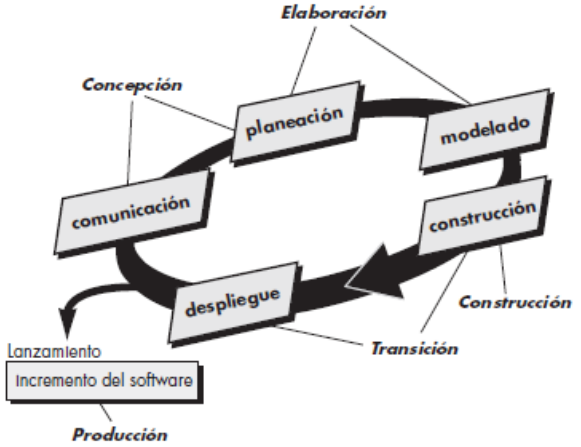


Consideraciones del modelo RUP:

- El proceso unificado es un intento por obtener los mejores rasgos y características de los modelos tradicionales del proceso del software, pero en forma que implemente muchos de los mejores principios del desarrollo ágil de software.
- El proceso unificado reconoce la importancia de la comunicación con el cliente y los métodos directos para describir su punto de vista respecto de un sistema (el caso de uso).
- El UML brinda la tecnología necesaria para apoyar la práctica de la ingeniería de software orientada a objetos.
- Actualmente, el proceso unificado (PU) y el UML se usan mucho en proyectos de toda clase orientado a objetos.
- El modelo iterativo e incremental propuesto por PU puede y debe adaptarse para que satisfaga necesidades específicas del proyecto.

Modelos de proceso especializado: Proceso Unificado Racional RUP

Fase	Descripción
1. Concepción	La meta de la fase de concepción es establecer un caso empresarial para el sistema. Deben identificarse todas las entidades externas **(personas y sistemas)** que interactuarán con el sistema y definirán dichas interacciones. Luego se usa esta información para **valorar la aportación** del sistema hacia la empresa. Si esta aportación es menor, entonces el proyecto puede cancelarse después de esta fase.
2. Elaboración	Las metas de esta fase consisten en desarrollar la comprensión del problema de dominio, establecer un **marco conceptual arquitectónico** para el sistema, diseñar el **plan del proyecto** e identificar los **riesgos** clave del proyecto. Al completar esta fase, debe tenerse un modelo de requerimientos para el sistema, que podría ser una serie de casos de uso del UML, una descripción arquitectónica y un plan de desarrollo para el software.
3. Construcción	La fase de construcción incluye diseño, programación y pruebas del sistema. Partes del sistema se desarrollan en paralelo y se integran durante esta fase. Al completar ésta, debe tenerse un sistema de software funcionando y la documentación relacionada y lista para entregarse al usuario.
4. Transición	La fase final del RUP se interesa por el cambio del sistema desde la comunidad de desarrollo hacia la comunidad de usuarios, y por ponerlo a funcionar en un ambiente real. Esto es algo ignorado en la mayoría de los modelos de proceso de software aunque, en efecto, es una actividad costosa y en ocasiones problemática. En el complemento de esta fase se debe tener un sistema de software documentado que funcione correctamente en su entorno operacional.
5. Producción	Durante esta fase, se vigila el uso que se da al software, se brinda apoyo para el ambiente de operación (infraestructura) y se reportan defectos y solicitudes de cambio para su evaluación



Modelos de proceso especializado: Proceso Unificado Racional RUP

La visión estática del RUP se enfoca en las actividades que tienen lugar durante el proceso de desarrollo, llamados **flujos de trabajo.** En el proceso se identifican 6 **flujos de trabajo de proceso centrales** y 3 *flujos de trabajo de apoyo centrales*. El RUP se diseñó en conjunto con el UML, de manera que la descripción del flujo de trabajo se orienta sobre modelos UML asociados.

Flujo de trabajo	Descripción
▪ Modelado del negocio	Se modelan los procesos de negocios utilizando casos de uso de la empresa
▪ Requerimientos	Se identifican los actores que interactúan con el sistema y se desarrollan casos de uso para modelar los requerimientos del sistema.
▪ Análisis y diseño	Se crea y documenta un modelo de diseño utilizando modelos arquitectónicos, de componentes, de objetos y de secuencias.
▪ Implementación	Se implementan y estructuran los componentes del sistema en subsistemas de implementación. La generación automática de código a partir de modelos de diseño ayuda a acelerar este proceso.
▪ Pruebas	Las pruebas son un proceso iterativo que se realiza en conjunto con la implementación. Las pruebas del sistema siguen al completar la implementación.
▪ Despliegue	Se crea la liberación de un producto, se distribuye a los usuarios y se instala en su lugar de trabajo.
▪ Administración de la configuración ▪ y del cambio	Este flujo de trabajo de apoyo gestiona los cambios al sistema.
▪ Administración del proyecto	Este flujo de trabajo de apoyo gestiona el desarrollo del sistema.
▪ Entorno	Este flujo de trabajo pone a disposición del equipo de desarrollo de software, las herramientas adecuadas de software.

Modelos de proceso especializado: Proceso Unificado Racional RUP

El enfoque práctico del RUP describe las **buenas prácticas de ingeniería de software** que se recomiendan para su uso en el desarrollo de sistemas. Las seis mejores prácticas fundamentales que se recomiendan son:

Mejores practicas	Descripción
1. Desarrollo de software de manera iterativa	Incrementar el plan del sistema con base en las prioridades del cliente, y desarrollar oportunamente las características del sistema de mayor prioridad en el proceso de desarrollo.
2. Gestión de requerimientos	Documentar de manera explícita los requerimientos del cliente y seguir la huella de los cambios a dichos requerimientos. Analizar el efecto de los cambios sobre el sistema antes de aceptarlos..
3. Usar arquitecturas basadas en componentes	Estructurar la arquitectura del sistema en componentes
4. Software modelado visualmente	Usar modelos **UML** gráficos para elaborar representaciones de software estáticas y dinámicas.
5. Verificar la calidad del software	Garantizar que el software cumpla con los estándares de calidad de la organización.
6. Controlar los cambios al software	Gestionar los cambios al software con un sistema de administración del cambio, así como con procedimientos y herramientas de administración de la configuración.

El RUP no es un proceso adecuado para todos los tipos de desarrollo, por ejemplo, para desarrollo de **software embebido**.

Modelos de proceso especializado: Personal o equipo de software

- El *proceso personal del software* (PPS) pone el énfasis en la medición personal tanto del producto del trabajo que se genera como de su calidad. Además, el PPS responsabiliza al profesional acerca de la planeación del proyecto (por ejemplo, estimación y programación de actividades) y delega en el practicante el poder de controlar la calidad de todos los productos del trabajo de software que se desarrollen.
- El PPS enfatiza la necesidad de detectar pronto los errores; de igual importancia es entender los tipos de ellos que es probable cometer. Esto se logra a través de una actividad de evaluación rigurosa ejecutada para todos los productos del trabajo que se generen.

Actividades	Descripción
1. Planeación	Esta actividad aísla los requerimientos y desarrolla las estimaciones tanto del tamaño como de los recursos. Además, realiza la estimación de los defectos (el número de defectos proyectados para el trabajo). Todas las mediciones se registran en hojas de trabajo o plantillas. Por último, se identifican las tareas de desarrollo y se crea un programa para el proyecto.
2. Diseño de alto nivel	Se desarrollan las especificaciones externas para cada componente que se va a construir y se crea el diseño de componentes. Si hay incertidumbre, se elaboran prototipos. Se registran todos los aspectos relevantes y se les da seguimiento.
3. Revisión de diseño de alto nivel	Se aplican métodos de verificación formal (véase el capítulo 21) para descubrir errores en el diseño. Se mantienen las mediciones para todas las tareas y resultados del trabajo importantes.
4. Desarrollo	Se mejora y revisa el diseño del componente. El código se genera, revisa, compila y prueba. Las mediciones se mantienen para todas las tareas y resultados de trabajo de importancia.
5. Post mortem	Se determina la eficacia del proceso por medio de medidas y mediciones obtenidas (ésta es una cantidad sustancial de datos que deben analizarse con métodos estadísticos). Las medidas y mediciones deben dar la guía para modificar el proceso a fin de mejorar su eficacia.

Modelos de proceso de software ágil

Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Tipos de Modelos	En que consiste
1. Método de desarrollo dinámico DSDM	Es un método que provee un framework para el desarrollo ágil de software, apoyado por su continua implicación del usuario en un desarrollo iterativo y creciente que sea sensible a los requerimientos cambiantes, para desarrollar un sistema que reúna las necesidades de la empresa en tiempo y presupuesto. Es uno de un número de métodos de desarrollo ágil de software y forma parte de la alianza ágil..
2. Programación extrema XP	Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.
3. Scrum	Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (clientes externos o internos), y el Teamque incluye a los desarrolladores.

SWEBOK 3.0

Contiene conceptos y conocimientos acerca de la ingeniería de software y de cómo debe de llevarse a cabo por un ingeniero de software, proporciona una visión general acerca del software de calidad y de las buenas prácticas de desarrollo, las áreas de conocimiento de han ido ampliando a través de las versiones.

Áreas principales

- Requisitos.
- Diseño.
- Desarrollo.
- Pruebas.
- Mantenimiento.
- Gestión de configuración.
- Gestión de software.
- Proceso de ingeniería.
- Herramientas y métodos de ingeniería.
- Calidad de software

Esteban, L. Rojas, W. Sánchez, M. (2013) Modelo de investigación en gestión de proyectos para la investigación en ingeniería. [Áreas de conocimiento para la ingeniería del software según el SWEBOK]. Recuperado de http://www.scielo.org.co/scielo.php?pid=S0120-81602013000100005&script=sci_arttext

SWEBOK 3.0

1

Requisito

Interpreta las necesidades del cliente en una lista de objetivos a cumplir, cada uno se puede trasformar en un subsistema o sólo en una funcionalidad, si los requisitos no son interpretados correctamente, el sistema sufrirá consecuencias graves en el desarrollo y mantenimiento

2

Diseño

Define la arquitectura, interfaces, diagramas de flujo, entre otros del sistema; en esta etapa se analizan los requerimientos y se crean posibles soluciones gráficas a cada uno.

3

Desarrollo

Interpreta la arquitectura, esquemas y diagramas de flujo definidos en la etapa de diseño en codificación de un lenguaje de programación, interactuando también con el sistema operativo y algunas veces con los dispositivos de entrada y salida.

4

Pruebas

Evalúa la eficiencia y calidad del producto detectando las posibles mejoras o fallas.

SWEBOK 3.0

5 Gestión de configuración

Identifica la configuración general del sistema para realizar posibles adaptaciones y configurar su ciclo de vida, detecta las características físicas y funcionales del sistema además del cumplimiento de sus objetivos.

7 Gestión del proceso

Valida todas las etapas del proceso, como las tareas que componen el proceso, funciones, mediciones, configuración, mantenimiento, entre otros.

6 Gestión de ingeniería

Verifica la infraestructura del proyecto, el control y la planeación del programa, asegura que el mantenimiento del producto sea adecuado.

8 Herramientas y procesos de ingeniería

son todos los recursos virtuales que nos ayudan a realizar tareas exhaustivas como, validar el producto, crear el diseño, realizar pruebas, detectar fallas, entre otros; pero es importante saber que estas herramientas sólo interpretan el resultado de la información que brindamos, si la información es errónea, el resultado puede llevarnos a una mala decisión.

9 Calidad de software

El conjunto de las actividades vistas anteriormente tiene como objetivo crear un producto de calidad, el cual pueda brindar al cliente la solución a los procesos que realiza o a la problemática que tiene, siendo eficaz, costeable, moldeable, y que pueda tener futuras implementación, éstas son algunas características de software de calidad.

Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.