PROCESO DE SOFTWARE

MSc Raúl Córdova Bayas
2022

2.1 El proceso de software

- 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

2.1 El proceso de software

Existen muchos diferentes procesos de software, pero todos deben incluir cuatro actividades que son fundamentales para la ingeniería de software:

1. *Especificación del software* Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.

2. *Diseño e implementación del software* Debe desarrollarse el software para cumplir con las especificaciones.

3. *Validación del software* Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.

4. *Evolución del software* El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

2.1 El proceso de software

- En cierta forma, tales actividades forman parte de todos los procesos de software.
- En la práctica éstas son actividades complejas en sí mismas e incluyen subactividades tales como la validación de requerimientos, el diseño arquitectónico, la prueba de unidad, etcétera.
- También existen actividades de soporte al proceso, como la documentación y el manejo de la configuración del software.
- Cuando los procesos se discuten y describen, por lo general se habla de actividades como especificar un modelo de datos, diseñar una interfaz de usuario, etcétera, así como del orden de dichas actividades

2.1 El proceso de software

- Al igual que las actividades, también las descripciones de los procesos deben incluir:

1. Productos, que son los resultados de una actividad del proceso. Por ejemplo, el resultado de la actividad del diseño arquitectónico es un modelo de la arquitectura de software.

2. Roles, que reflejan las responsabilidades de la gente que interviene en el proceso. Ejemplos de roles: gerente de proyecto, gerente de configuración, programador, etcétera.

3. Precondiciones y postcondiciones, que son declaraciones válidas antes y después de que se realice una actividad del proceso o se cree un producto. Por ejemplo, antes de comenzar el diseño arquitectónico, una precondición es que el cliente haya aprobado todos los requerimientos; después de terminar esta actividad, una postcondición podría ser que se revisen aquellos modelos UML que describen la arquitectura.

2.1 El proceso de software

- Los procesos de software son complejos y, como todos los procesos intelectuales y creativos, se apoyan en personas con capacidad de juzgar y tomar decisiones.
- No hay un proceso ideal; además, la mayoría de las organizaciones han diseñado sus propios procesos de desarrollo de software.
- Los procesos han evolucionado para beneficiarse de las capacidades de la gente en una organización y de las características específicas de los sistemas que se están desarrollando.
- Para algunos sistemas, como los sistemas críticos, se requiere de un proceso de desarrollo muy estructurado.
- Para los sistemas empresariales, con requerimientos rápidamente cambiantes, es probable que sea más efectivo un proceso menos formal y flexible.

2.1 El proceso de software

- En ocasiones, los procesos de software se clasifican como dirigidos por un plan (*plandriven*) o como procesos ágiles.
- Los procesos dirigidos por un plan son aquellos donde todas las actividades del proceso se planean por anticipado y el avance se mide contra dicho plan.
- En los procesos ágiles, la planeación es incremental y es más fácil modificar el proceso para reflejar los requerimientos cambiantes del cliente.
- Como plantean Boehm y Turner (2003), cada enfoque es adecuado para diferentes tipos de software.
- Por lo general, uno necesita encontrar un equilibrio entre procesos dirigidos por un plan y procesos ágiles.

2.1 El proceso de software

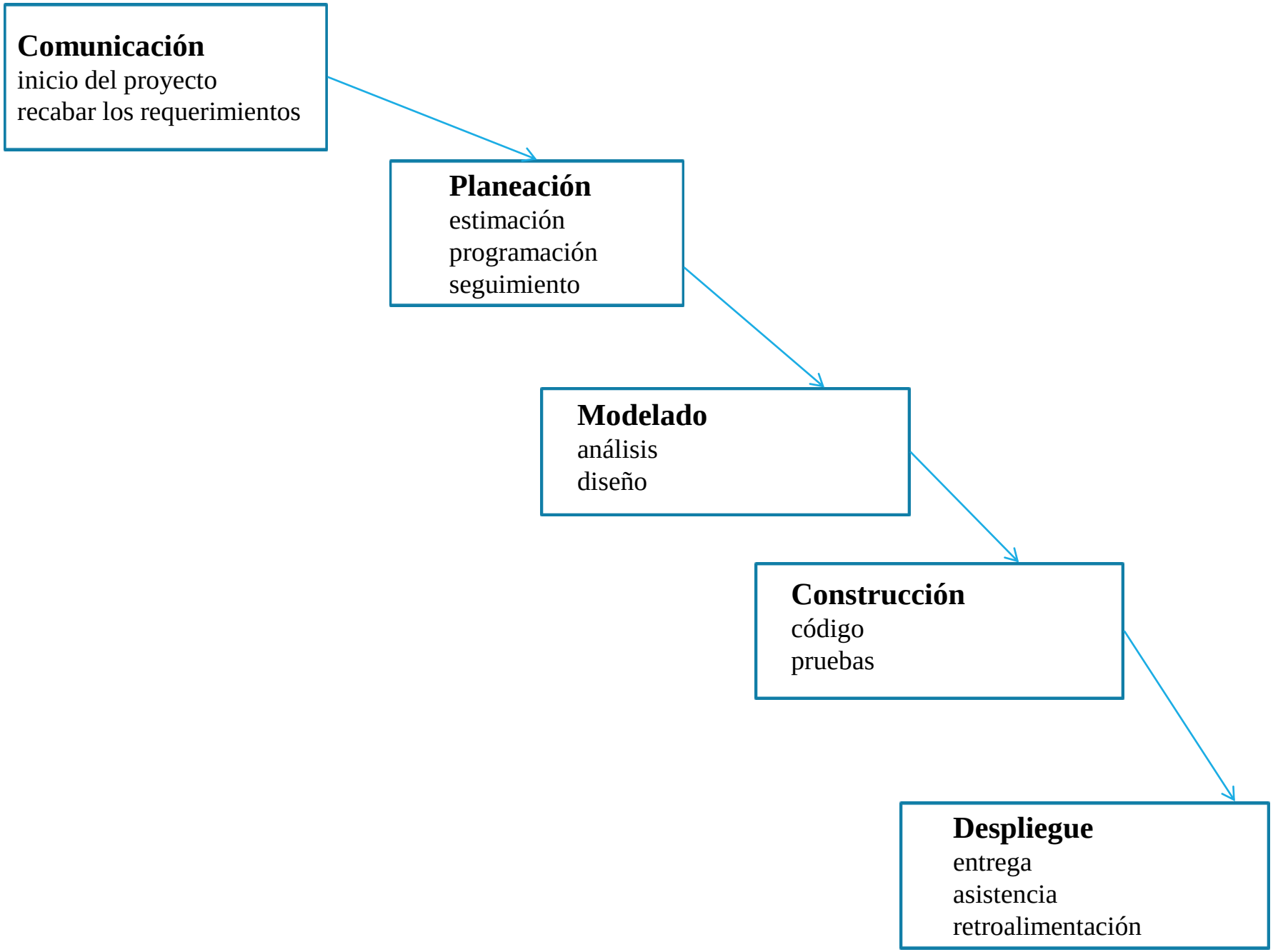
- Aunque no hay un proceso de software “ideal”, en muchas organizaciones sí existe un ámbito para mejorar el proceso de software.
- Los procesos quizás incluyan técnicas obsoletas o tal vez no aprovechen las mejores prácticas en la industria de la ingeniería de software.
- Hasta hoy, muchas organizaciones aún no sacan ventaja de los métodos de la ingeniería de software en su desarrollo de software.

2.1 El proceso de software

- Los procesos de software pueden mejorarse con la estandarización de los procesos, donde se reduce la diversidad en los procesos de software en una organización.
- Esto conduce a mejorar la comunicación, a reducir el tiempo de capacitación, y a que el soporte de los procesos automatizados sea más económico.
- La estandarización también representa un primer paso importante tanto en la introducción de nuevos métodos y técnicas de ingeniería de software, como en sus buenas prácticas.

2.2 Modelos del proceso de software

- MODELO CASCADA
  - Se aplica cuando los requerimientos del problema se comprenden bien.
  - Cuando el trabajo desde la **comunicación** hasta el **despliegue** fluye en forma razonablemente lineal.
  - Esto se puede dar cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente.
  - Ejemplo:
    - la adaptación para un software de contabilidad que es obligatorio hacer debido a cambios en las regulaciones gubernamentales.
  - También ocurre en cierto número limitado de nuevos esfuerzos de desarrollo, pero sólo cuando los requerimientos están bien definidos y tienen una estabilidad razonable.


Comunicación
inicio del proyecto
recabar los requerimientos
Planeación
estimación
programación
seguimiento
Modelado
análisis
diseño
Construcción
código
pruebas
Despliegue
entrega
asistencia
retroalimentación

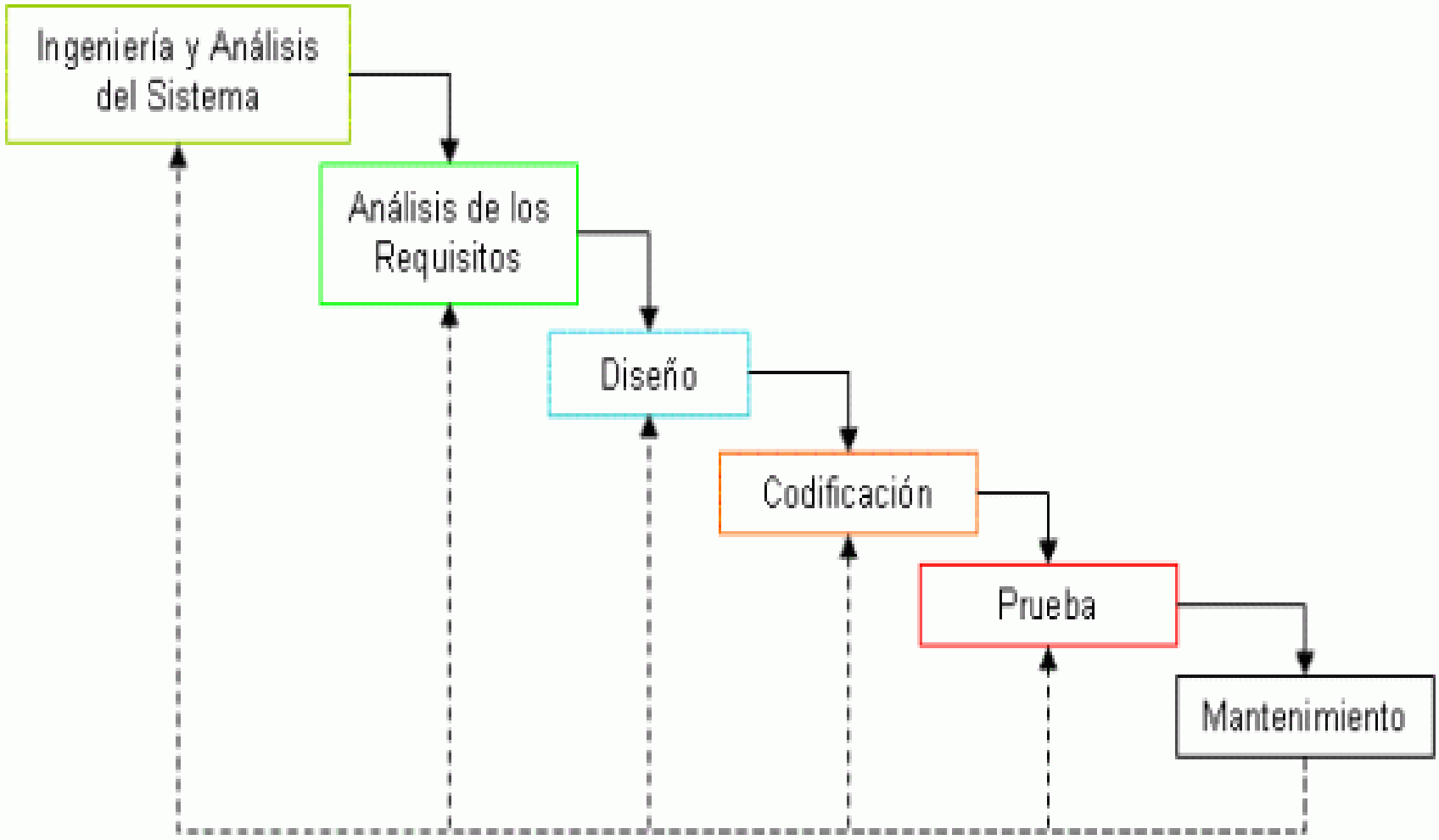


fig 4 : Modelo Cascada

Modelo en cascada (cont.)

- También se llama *ciclo de vida clásico*.
- Sugiere un enfoque sistemático y secuencial para el desarrollo del software.
- Comienza con la especificación de los requerimientos por parte del cliente.
- Avanza a través de planeación, modelado, construcción y despliegue.
- Concluye con el apoyo del software terminado (asistencia).
- Una variante se denomina *modelo en V*.
- Este modelo muestra la relación entre las acciones para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana.

Modelo en V

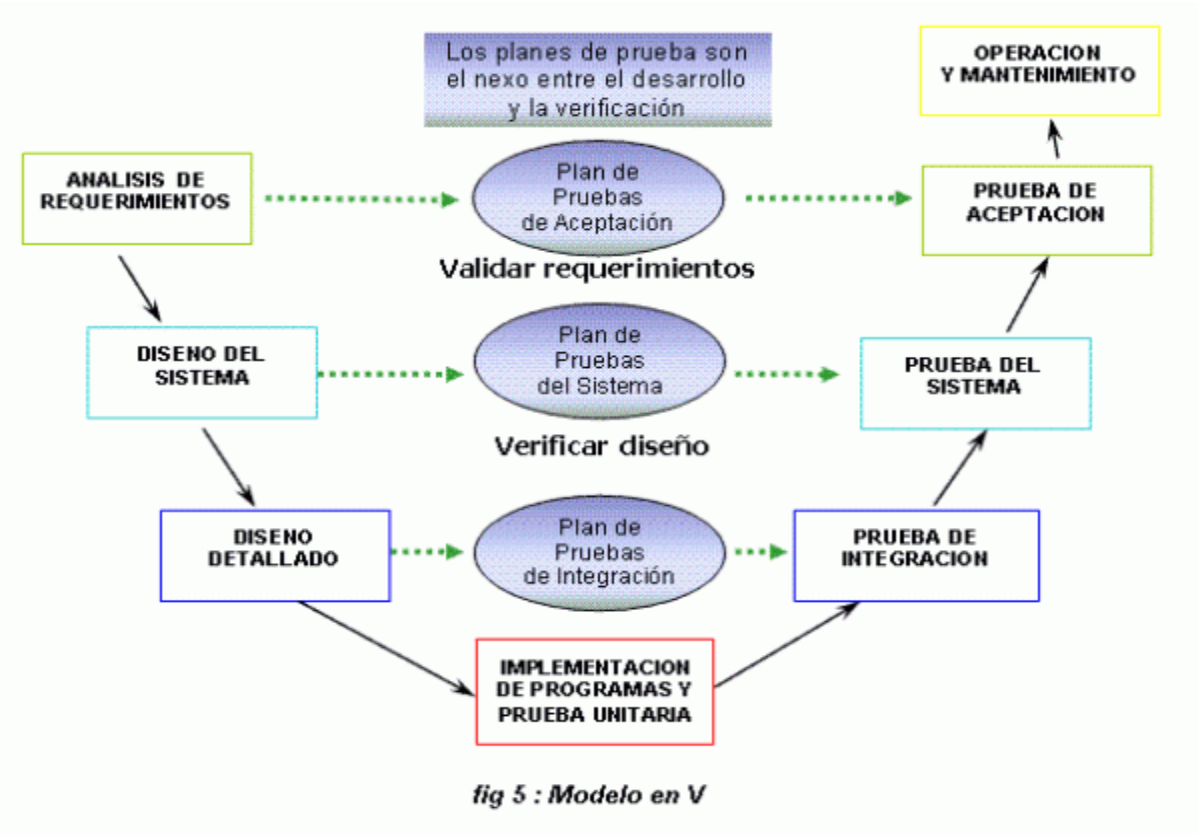


fig 5 : Modelo en V

Modelo en V (cont.)

- A medida que se avanza hacia abajo por el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas.
- Una vez generado el código, el equipo sube por el lado derecho de la V ejecutando una serie de pruebas.
- Las pruebas o acciones para asegurar la calidad validan cada uno de los modelos creados cuando se pasó por el lado izquierdo de la V.
- Los modelos cascada y en V no tienen diferencias.
- El modelo en V proporciona una forma de visualizar el modo de aplicación de las acciones de verificación y validación al trabajo de ingeniería inicial.

Modelo en cascada: problemas

- Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo:
  - Aunque se aceptan repeticiones, éstas se las hace indirectamente.
  - Los cambios generan confusión conforme se avanza en el proyecto.
- A menudo es difícil para el cliente enunciar todos los requerimientos en forma explícita y desde el principio.
- El cliente debe tener paciencia:
  - No se dispondrá de una versión funcional del sistema hasta que el proyecto esté muy avanzado.
  - Un error grande sería desastroso si se lo detecta solamente en la fase de pruebas.
- Se generan estados de bloqueo entre equipos.

Modelo en cascada: ventajas y usos

- Es un modelo sistemático y ordenado para generar software
- Genera una documentación completa del proyecto.
- Una etapa se inicia solamente si la anterior se termina.
- Permite hacer un control de calidad en cada hito del proyecto.
- Es útil en situaciones en las que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final.
- Apropiado para sistemas muy complejos en que los requerimientos no cambian desde el principio:
  - Problemas de ingeniería y científicos que ameritan el trabajo de varios equipos, inclusive ubicados en diferentes lugares geográficos.

Modelo de proceso incremental

- Usado cuando los requerimientos del software están razonablemente bien definidos, pero el alcance del esfuerzo de desarrollo imposibilita un proceso lineal.
- También cuando hay la necesidad imperiosa de dar rápidamente cierta funcionalidad limitada a los usuarios y aumentarla en entregas posteriores.
- Combina elementos de los flujos de proceso lineal y paralelo.
- Aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades del proyecto.
- Cada secuencia produce incrementos de software susceptibles de entregarse de manera similar a los incrementos producidos en un flujo de proceso evolutivo, que se lo verá posteriormente.

Modelo de proceso incremental (cont.)

- El flujo de proceso para cualquier incremento puede incorporar el paradigma del prototipo.
- Es frecuente que el primer incremento sea el *producto fundamental*:
  - Se abordan los requerimientos básicos, pero no se proporcionan muchas características suplementarias (algunas conocidas y otras no).
  - El cliente usa el producto fundamental o lo somete a una evaluación detallada.
  - En función de esta evaluación se desarrolla un plan para el incremento que sigue.
  - El plan incluye la modificación del producto fundamental para cumplir con las necesidades del cliente, así como la entrega de características adicionales y más funcionalidad.
- El proceso se repite después de entregar cada incremento, hasta terminar el producto final.

Modelo de proceso incremental

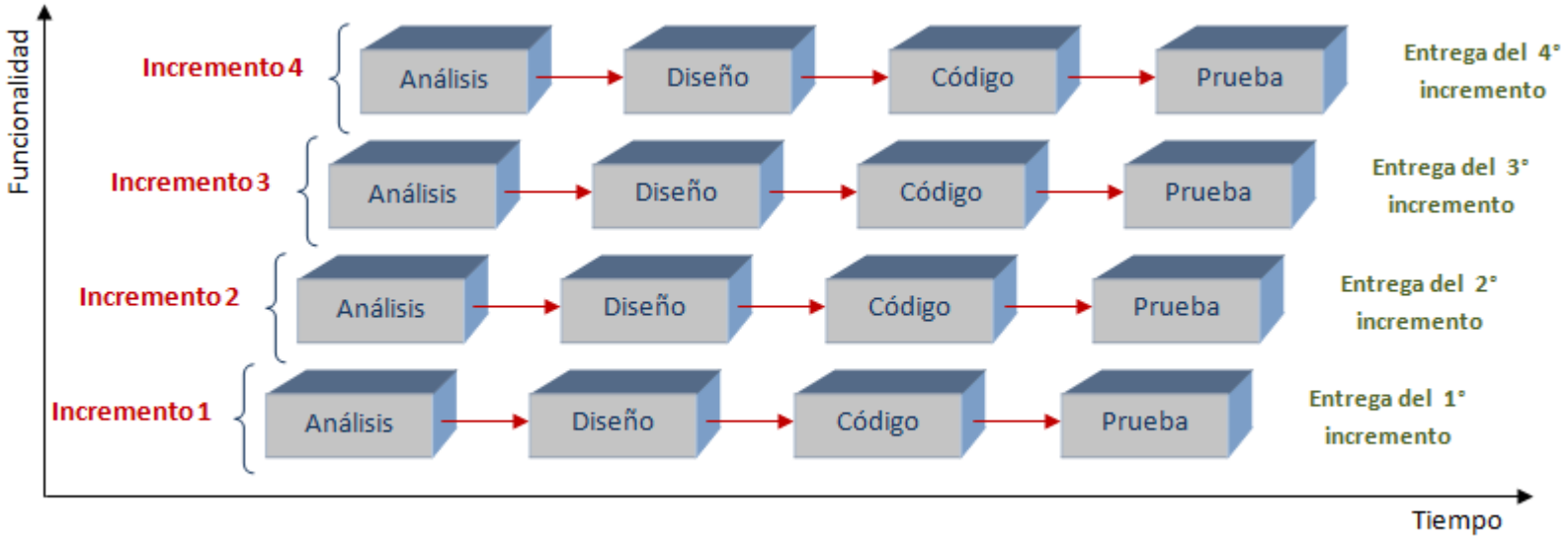


Figura 1: El Modelo Incremental

Modelo de proceso incremental (cont.)

- El modelo se centra en que en cada incremento se entrega un producto que ya opera.
- Es útil cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido.
- Los primeros incrementos se desarrollan con poco personal.
- Si el producto básico es bien recibido, se agrega más personal (si se requiere) para que labore en el siguiente incremento.
- Los incrementos se planean para administrar riesgos técnicos.

Modelo de proceso incremental: ventajas y desventajas

- VENTAJAS:
  - Tiene participación permanente del usuario.
  - Permite cambio de requerimientos a lo largo del proceso.
  - Permite entregas funcionales del sistema por cada incremento.
  - Está ideado para manejar los incrementos en función de los riesgos del proyecto.
- DESVENTAJAS:
  - Con muchos incrementos se puede perder la visión del producto final.
  - La integración es particularmente importante en cada incremento.

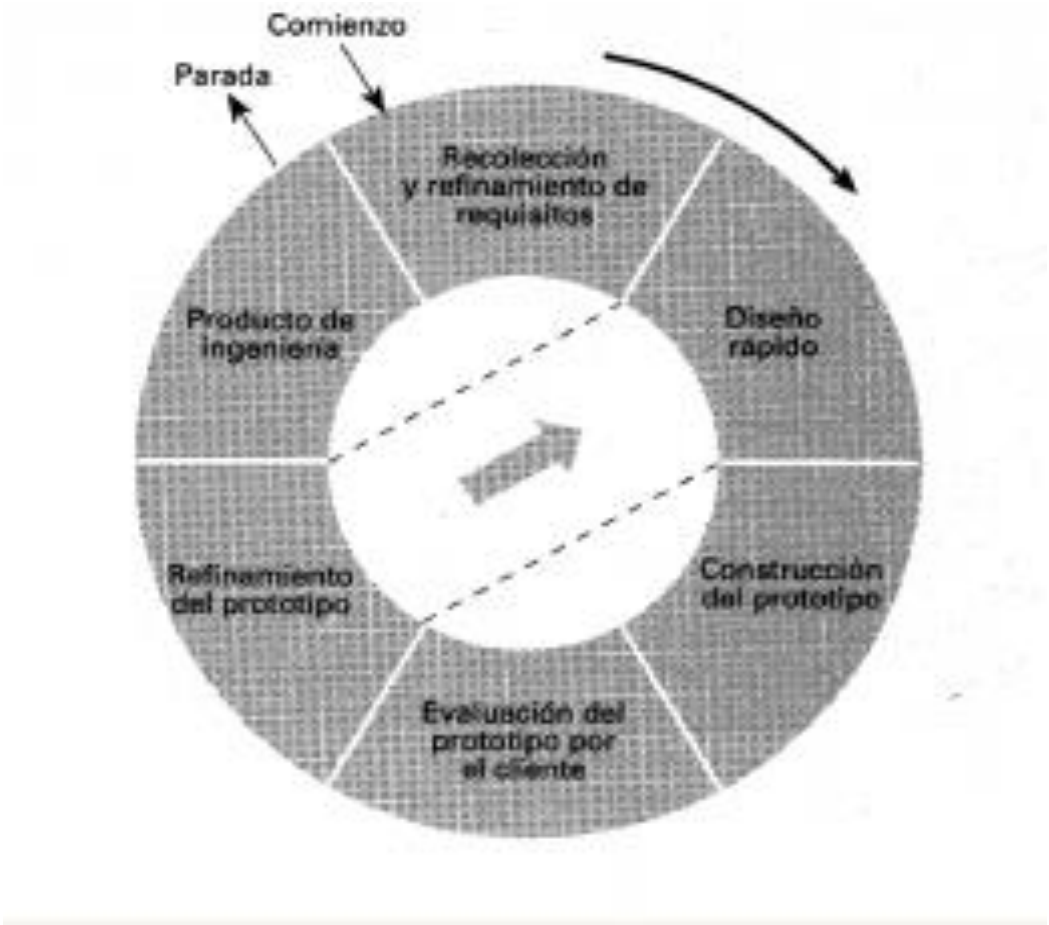
Modelos de procesos evolutivos

- Los requerimientos del negocio y del producto cambian en el tiempo.
- Los plazos apretados del mercado hacen imposible la terminación de un software perfecto.
- Por lo que debe lanzarse una versión limitada a fin de aliviar la presión de la competencia o del negocio.
- Se comprenden bien los requerimientos pero los detalles del producto o extensiones del sistema están aún por definirse.
- Se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo.
- Estos son los modelos evolutivos que son iterativos.
- Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software.

Modelos de procesos evolutivos: prototipos

- Se lo puede usar como una técnica que puede implementarse en el contexto de cualquiera de los modelos de proceso descritos antes.
- Este paradigma ayuda a mejorar la comprensión de lo que hay que elaborar cuando los requerimientos no están claros.
- También puede usarse para determinar la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debe adoptar una interfaz de usuario.
- También puede usarse el modelo de prototipos como un modelo de proceso aislado.
- En este último caso, sirve fundamentalmente para desarrollar sistemas de software pequeños.

Modelo de prototipos



Modelos de procesos evolutivos: prototipos (cont.)

- Este paradigma comienza con la comunicación cuando se reúnen los participantes para:
  - Definir los objetivos generales del software,
  - Identificar cualesquiera de los requerimientos que se conozcan y
  - Detectar las áreas en las que es imprescindible una mayor definición.
- Luego se planea rápidamente una iteración para hacer el prototipo y se lleva a cabo el modelado, en la forma de un «diseño rápido».
- El diseño rápido se centra en la representación de aquellos aspectos del software que serán visibles para los usuarios finales (interfaces de usuario).
- El diseño rápido lleva a la construcción del prototipo.
- Este es entregado y evaluado por el usuario, quien da retroalimentación para mejorar o ampliar los requerimientos.

Modelos de procesos evolutivos: prototipos (cont.)

- La iteración ocurre a medida que el prototipo es afinado para satisfacer las necesidades de distintos participantes, permitiendo un entendimiento mejor de lo que se necesita hacer.
- El uso mayor del prototipo se da para identificar los requerimientos del sistema software.
- También se lo puede usar para probar herramientas, o para generar rápidamente código (usando generadores de reportes y administradores de ventanas) que permiten generar rápidamente programas que funcionen.
- Existen prototipos desechables, que se los deja de usar una vez cumplidos los objetivos para los cuales fue construido.
- También existen los prototipos evolutivos, que poco a poco se transforman en el sistema real.

Modelos de procesos evolutivos: prototipos (cont.)

DESVENTAJAS:

1. Falta de calidad general del software:
   - Generada por la prisa en hacer que funcionara.
   - Puede generarse un producto difícil de mantener.
   - Los clientes consideran al prototipo como el producto final.
   - El gerente de desarrollo cede ante las demandas de los clientes.
2. Compromisos respecto de la implementación:
   - Sistema operativo inapropiado.
   - Lenguaje de programación conocido o porque es el único con el que cuenta.
   - Algoritmo ineficiente sólo para demostrar capacidad.
   - Puede llevar a comodidad en las elecciones, olvidando las razones por las que eran inadecuadas.

Modelos de procesos evolutivos: prototipos (cont.)

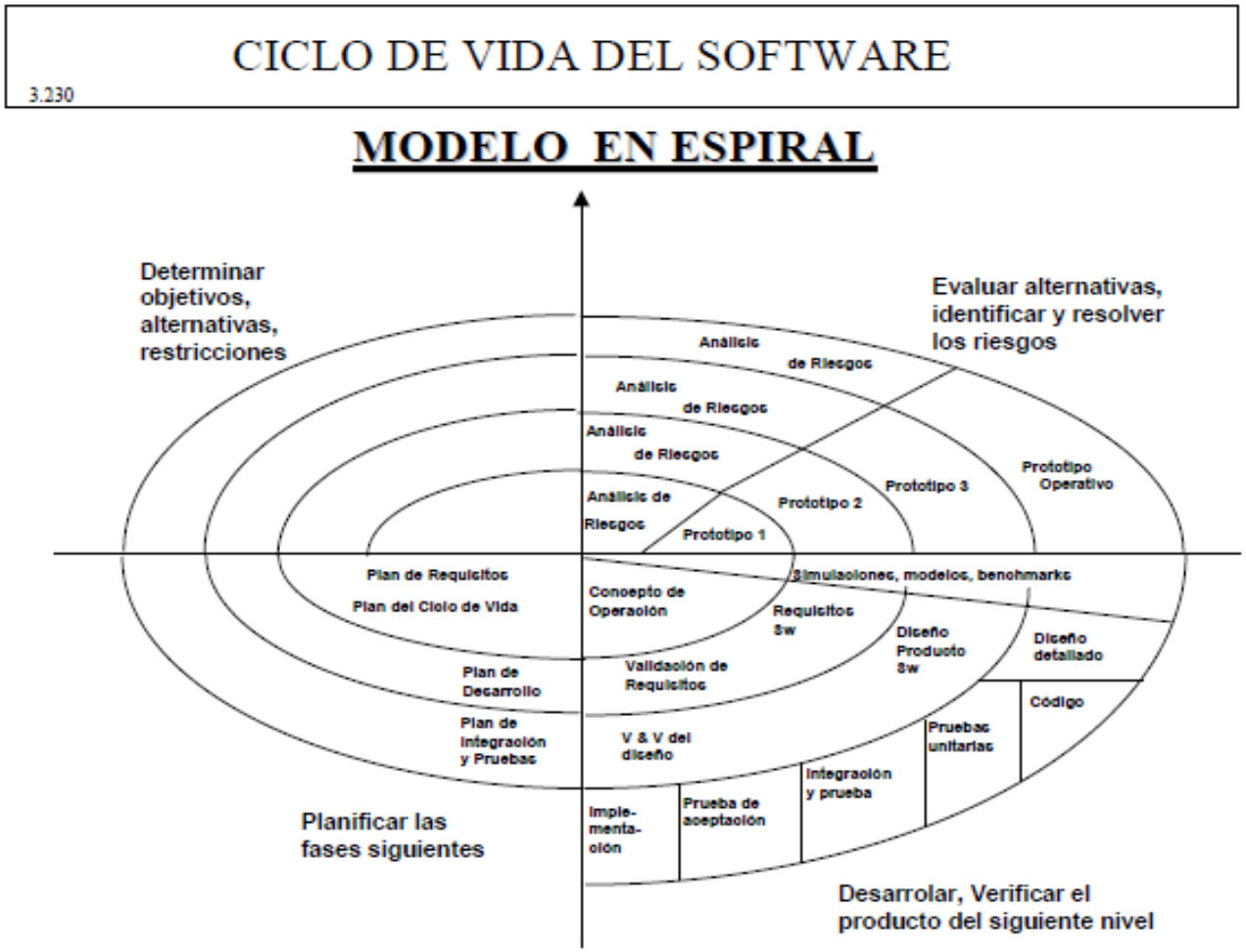
VENTAJAS:

1. Participación permanente del cliente
2. Rapidez en la elaboración del producto
3. No se requieren tener todos los requerimientos desde el inicio
4. Facilidad para hacer cambios durante el proceso.

Modelos de procesos evolutivos: Modelo Espiral

- Fue propuesto por Barry Boehm en 1988.
- Combina la naturaleza iterativa de los prototipos con los aspectos controlados y sistemáticos del modelo en cascada.
- Permite hacer un desarrollo rápido de versiones cada vez más completas.
- Es un modelo que se basa en los riesgos del proceso.
- El software se desarrolla en una serie de entregas evolutivas.
- En las primeras iteraciones lo que se entrega puede ser un modelo o prototipo.
- En las iteraciones posteriores se producen versiones cada vez más completas del sistema.

Modelo espiral



Modelos de procesos evolutivos: modelo espiral (cont.)

- El proceso comienza en el centro y sigue un sentido horario.
- El riesgo se considera conforme se desarrolla cada vuelta.
- En la primera vuelta se desarrolla una especificación del producto.
- En las vueltas siguientes se desarrolla un prototipo y luego versiones cada vez más completas del producto final.
- Cada vez que se pasa por planeación se hacen ajustes al plan del proyecto.
- El costo y el cronograma se ajustan en base a la retroalimentación obtenida del cliente después de la entrega para su evaluación.

Modelos de procesos evolutivos: modelo espiral (cont.)

- El modelo puede adaptarse para aplicarse a lo largo del ciclo de vida del software.
- En este caso, el primer circuito puede representar un «proyecto de desarrollo del concepto» que comienza en el centro de la espiral y continúa por iteraciones múltiples hasta terminar el desarrollo del concepto.
- Si el concepto va a desarrollarse en un producto real, el proceso sigue hacia fuera de la espiral y comienza un «proyecto de desarrollo de producto nuevo».
- Este producto nuevo evolucionará a través de varias iteraciones.
- Luego puede aplicarse el circuito para que represente un «proyecto de mejora del producto».
- Así, la espiral se aplica hasta que se retira el producto.

Modelos de procesos evolutivos: modelo espiral (cont.)

VENTAJAS:

- Este modelo es un enfoque realista para el desarrollo de sistemas y de software a gran escala.
- Se comprende y se reacciona mejor ante los riesgos en cada nivel de evolución.
- Usa los prototipos como mecanismo de reducción de riesgos.
- Permite aplicar el enfoque de hacer prototipos en cualquier etapa de la evolución del producto.
- Sigue el modelo cascada, pero incorporado en una estructura iterativa que refleja al mundo real en una forma más realista.
- Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto, esperando reducirlos antes que se vuelvan un problema.

Modelos de procesos evolutivos: modelo espiral (cont.)

DESVENTAJAS:

- Es difícil convencer a los clientes de que el enfoque evolutivo en espiral es controlable: se sabe cuándo comienza pero no se sabe cuándo termina.
- Demanda mucha experiencia en la evaluación del riesgo y se basa en ella para alcanzar el éxito del proyecto.
- Existen problemas cuando no se detectan riesgos y no se los administra.
- Es útil solamente para grandes proyectos.

Modelos de procesos evolutivos: conclusiones

- Son idóneos para sistemas que cambian continuamente, que requieren tiempos de entrega muy apretados y una necesidad apremiante de la satisfacción del cliente o usuario.
- En muchos casos, el tiempo para llegar al mercado es el requerimiento administrativo más importante.
- Si se pierde nicho de mercado, todo el proyecto podría carecer de sentido.

Modelos de procesos evolutivos: conclusiones

- Estos modelos, sin embargo, tienen demasiadas debilidades:

1. Hacer prototipos o seguir el proceso en espiral plantea un problema para la planeación del proyecto debido a la incertidumbre en el número de ciclos que se requieren para elaborar el producto: La mayor parte de técnicas de administración y estimación de proyectos se basa en un planteamiento lineal de las actividades, por lo que no se ajustan por completo.

2. Los procesos evolutivos no establecen la velocidad máxima de la evolución. Si ocurren demasiado rápido, sin un periodo de relajamiento, es seguro que el proceso se volverá un caos. Pero si la velocidad es muy lenta, se verá perjudicada la productividad.

Modelos de procesos evolutivos: conclusiones

- Debilidades:

3. Los procesos de software deben centrarse en la flexibilidad y capacidad de extensión en lugar de alta calidad. Sin embargo, debe darse prioridad a la velocidad del desarrollo con el enfoque de cero defectos. Extender el desarrollo a fin de lograr alta calidad podría dar como resultado la entrega tardía del producto, cuando haya desaparecido el nicho de oportunidad. Este cambio de paradigma es impuesto por la competencia al borde del caos.

Modelos de procesos evolutivos: conclusiones

- El objetivo de los modelos evolutivos es desarrollar software de alta calidad en forma iterativa e incremental.
- También se puede usar un proceso evolutivo para hacer énfasis en la flexibilidad, expansibilidad y velocidad del desarrollo.
- El reto es establecer un balance apropiado entre estos parámetros críticos del proyecto y el producto, y la satisfacción del cliente (que determina finalmente la calidad del software).