

INGENIERÍA DE SOFTWARE Y REQUERIMIENTOS

FACULTAD DE INGENIERÍA EN SISTEMAS

AGENDA

- El proceso de software
- Modelos del proceso de software
- Actividades del proceso de software
- Ejercicio Práctico
- Laboratorios
- Bibliografía
- Información adicional

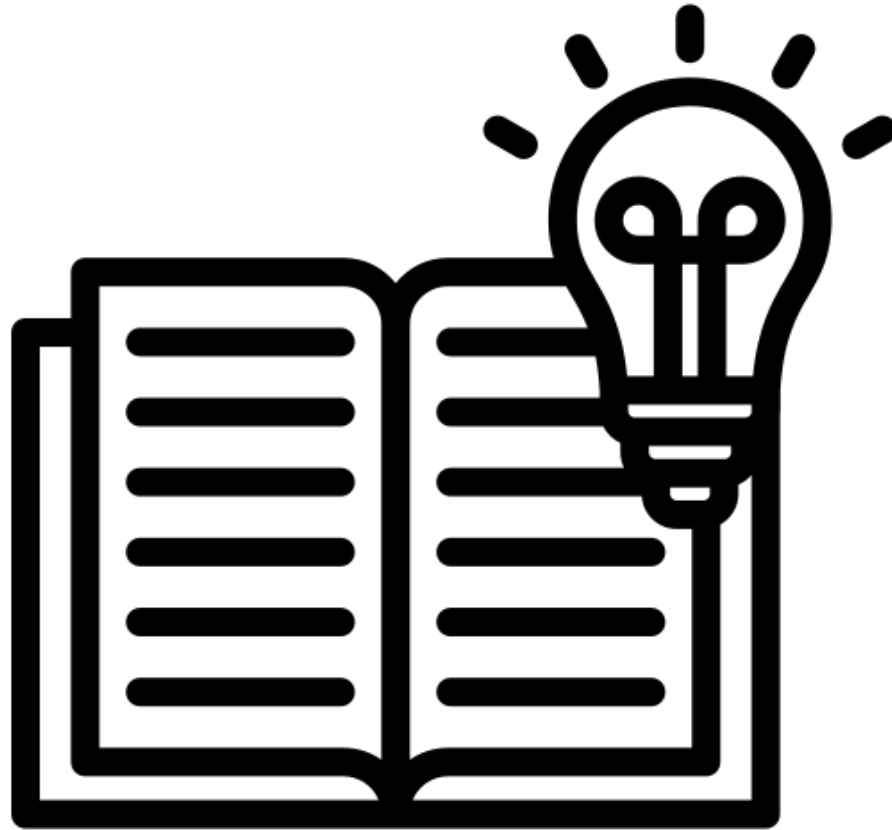
Capítulos y Subcapítulos	Descripción
Capítulo 1:	Software e Ingeniería de Software
1.1	La naturaleza del software
1.2	Conceptos básicos de software
1.3	Definición de Ingeniería de Software
1.4	Dominios de aplicación del software
Capítulo 2:	Proceso de Software
2.1	El proceso de software
2.2	Modelos del proceso de software
2.3	Actividades del proceso de software
Capítulo 3:	El Proceso Unificado
3.1	Introducción al proceso unificado de desarrollo de software
3.2	Las cuatro P en el desarrollo de software
3.3	Dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental
3.4	Los flujos de trabajo fundamentales
Capítulo 4:	Ingeniería de Requerimientos
4.1	Definición, niveles y propiedades de los requerimientos
4.2	Requerimientos desde la perspectiva del cliente
4.3	Requerimientos del negocio, usuario, software y reglas del negocio
4.4	El Ingeniero de Requerimientos / Analista de Negocios
Capítulo 5:	Elicitación y Análisis de Requerimientos
5.1	Actividades de elicitación de requerimientos
5.2	Técnicas de elicitación de requerimientos
5.3	Análisis de requerimientos
Capítulo 6:	Especificación y Validación de Requerimientos
6.1	Documentación de requerimientos
6.2	Escritura de requerimientos
6.3	Validación de requerimientos
6.4	El estándar IEEE 830



OBJETIVO DE LA CLASE

- Al finalizar la clase, los estudiantes serán capaces de comprender las características fundamentales de los procesos del software y su aplicabilidad en el ciclo de vida del desarrollo de software.
- Comprenderá los conceptos y modelos sobre procesos de software
- Comprender acerca de los modelos de proceso de software y sabrá cuándo usarlos.
- Entender las principales actividades del proceso de ingeniería de requerimientos de software.

INTRODUCCIÓN TEÓRICA



PROCESOS 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# o cualquier otro.
- 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.

PROCESOS DE SOFTWARE

- Un **proceso** es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo.
- Una **actividad** busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software.
- Una **acción** (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura).
- Una **tarea** se centra en un objetivo pequeño pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible.

ESTRUCTURA DEL PROCESOS DE SOFTWARE

- Proceso

└─• Actividad

└─• Acciones

└─• Tarea

Proceso del software

Estructura del proceso

Actividades sombrilla

actividad estructural # 1

acción de ingeniería de software # 1.1

Conjuntos
de tareas

⋮

acción de ingeniería de software # 1.k

Conjuntos
de tareas

⋮

actividad estructural # n

acción de ingeniería de software # n.1

Conjuntos
de tareas

⋮

acción de ingeniería de software # n.m

Conjuntos
de tareas

tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto

tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto

tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto

tareas del trabajo
productos del trabajo
puntos de aseguramiento de la calidad
puntos de referencia del proyecto

PROCESOS DE SOFTWARE

- En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo.
- Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

PROCESOS DE SOFTWARE

- La estructura del proceso establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad.

PROCESOS DE SOFTWARE

Según Ian Sommerville, existen diferentes **procesos de software**, pero todos al menos deben incluir cuatro **actividades** que son **fundamentales** para la ingeniería de software:

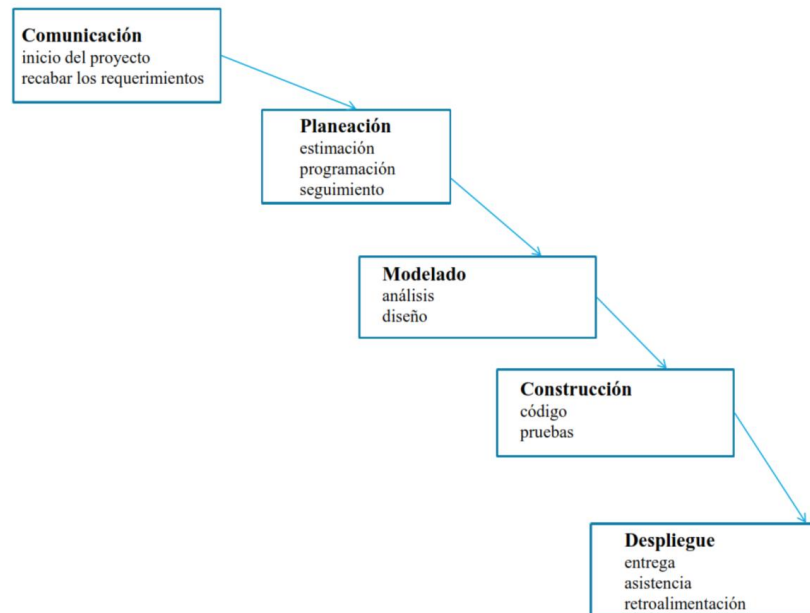
- **Especificación** del software
 - Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.
- **Diseño** e implementación del software
 - Debe desarrollarse el software para cumplir con las especificaciones.
- **Validación** del software
 - Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.
- **Evolución** del software
 - El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

PROCESOS DE SOFTWARE

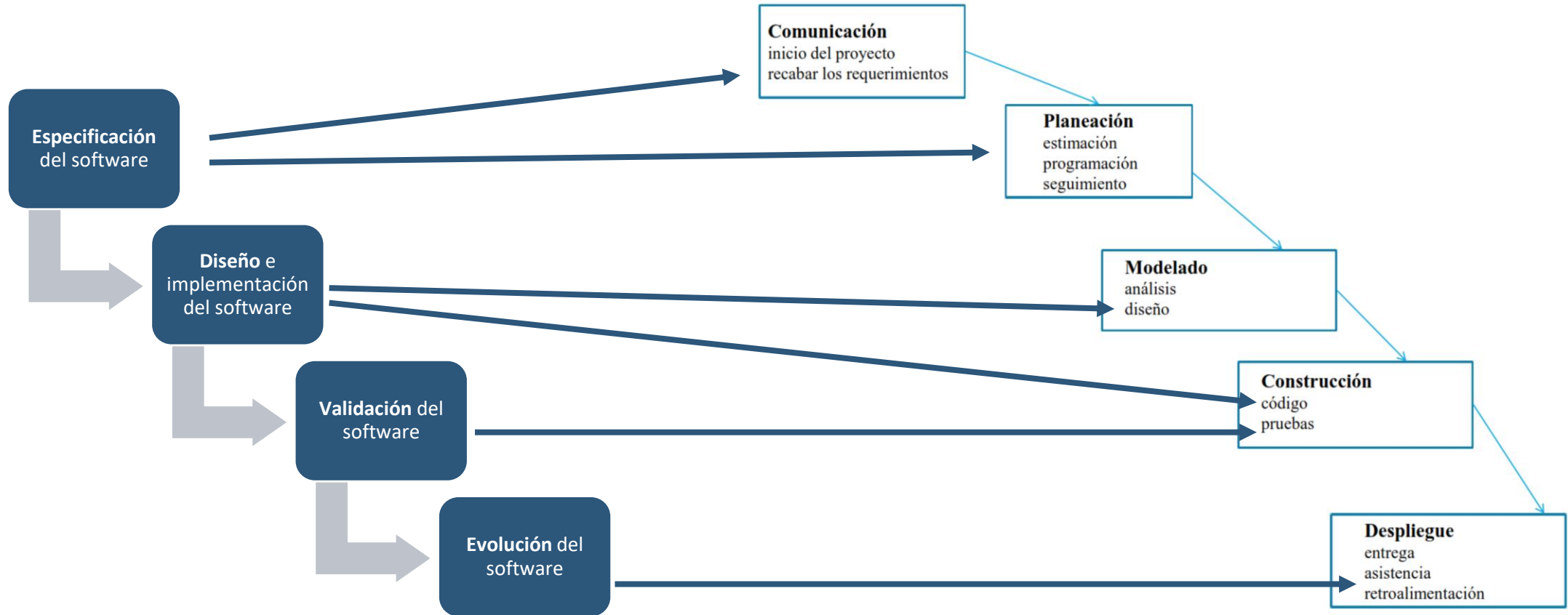
Además, la estructura del proceso de software incluye un conjunto de actividades sombrilla que son aplicables a través de todo el proceso del software.

Según Roger Pressman, Una estructura de **proceso general** para la ingeniería de software consta de cinco actividades:

1. Comunicación
2. Planeación
3. Modelado
4. Construcción
5. Despliegue



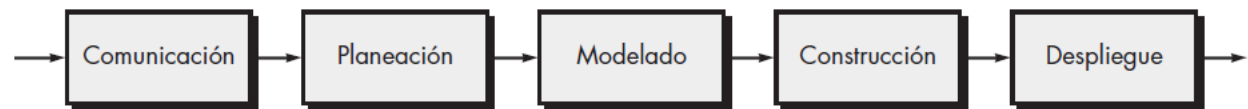
FLUJO DEL PROCESOS DE SOFTWARE



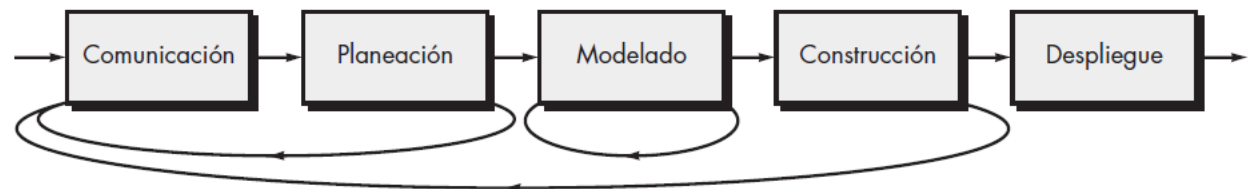
Ian Sommerville, [procesos de software](#)

Roger Pressman, estructura de [proceso general](#)

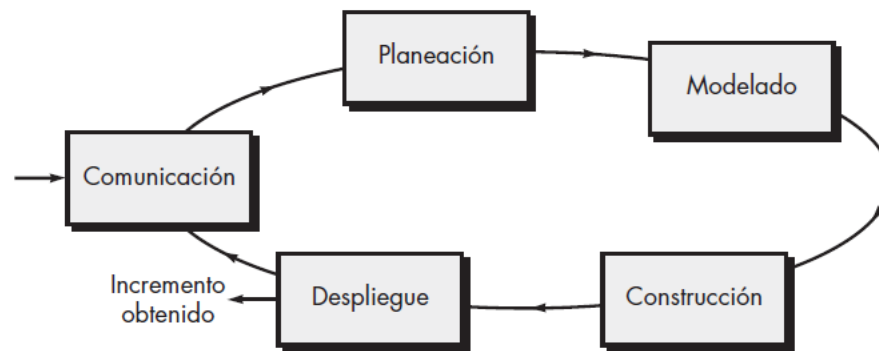
FLUJO DEL PROCESOS DE SOFTWARE



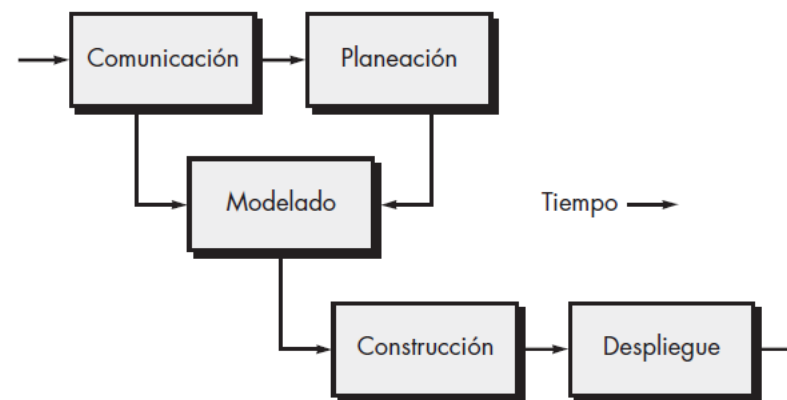
a) Flujo de proceso lineal



b) Flujo de proceso iterativo



c) Flujo de proceso evolutivo

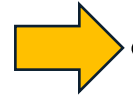


d) Flujo de proceso paralelo

MODELOS DEL PROCESO DE SOFTWARE

- Modelo en Cascada
- Modelo en V
- Modelo de proceso incremental
- Modelos de procesos evolutivos
 - Modelo de prototipos
 - Modelo espiral

MODELOS DEL PROCESO DE SOFTWARE



- Modelo en Cascada
- Modelo en V
- Modelo de proceso incremental
- Modelos de procesos evolutivos
 - Modelo de prototipos
 - Modelo espiral

EL MODELO EN CASCADA (WATERFALL)

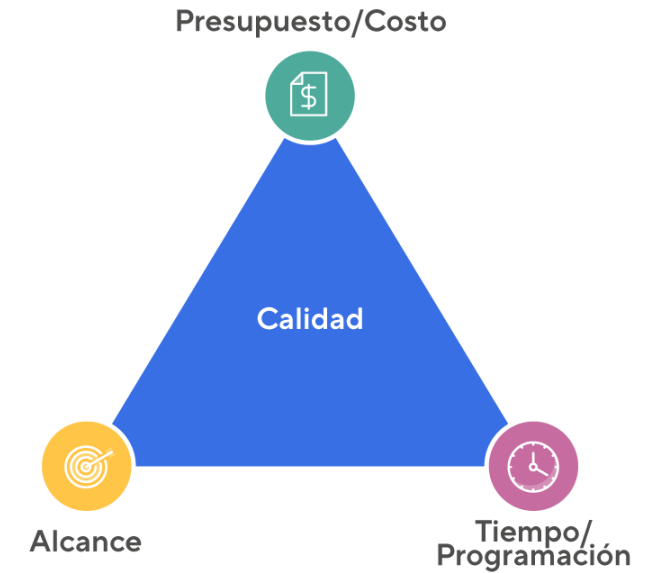
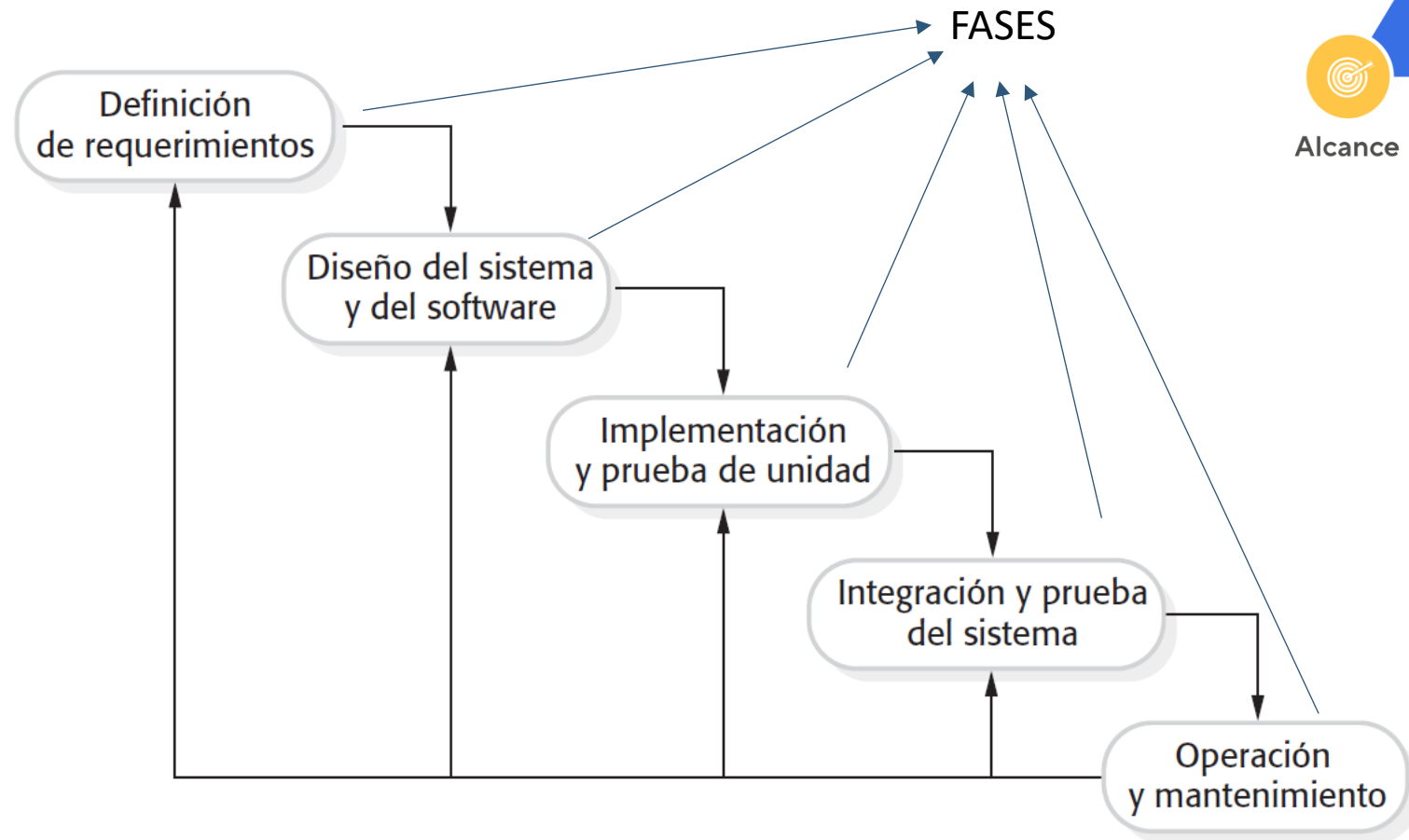
- Se aplica cuando los requerimientos del problema se comprenden bien.
- Cuando el trabajo desde la comunicación hasta el despliegue fluye en forma razonablemente lineal.
- Esto se puede dar cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente.
- Ejemplo:
 - la adaptación 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.

EL MODELO EN CASCADA (WATERFALL)

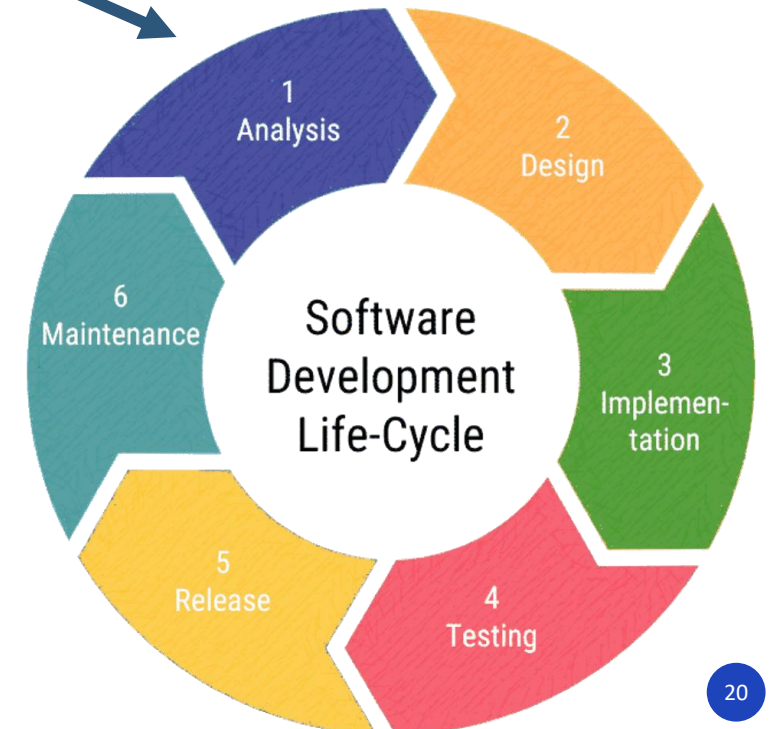
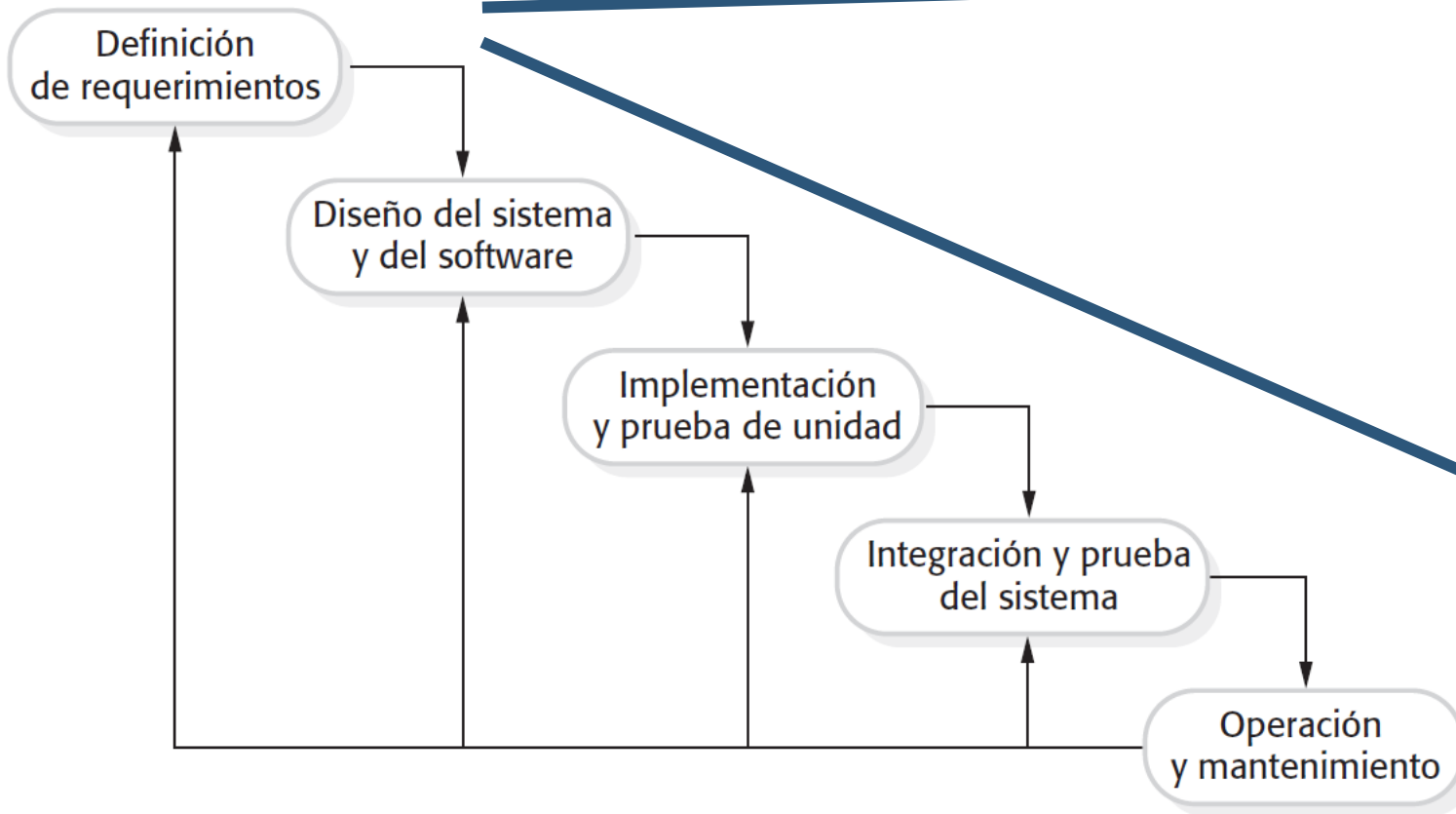
Consta de varias fases

- Llamado así por la posición de las fases en el desarrollo de esta, que parecen caer en cascada “por gravedad” hacia las siguientes fases
- Es un enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.
- Al final de cada etapa, se lleva a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.
- Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida.
- Fue propuesta inicialmente por Winston W. Royce en 1970 y posteriormente revisada por Barry Boehm en 1980 e Ian Sommerville en 1985.

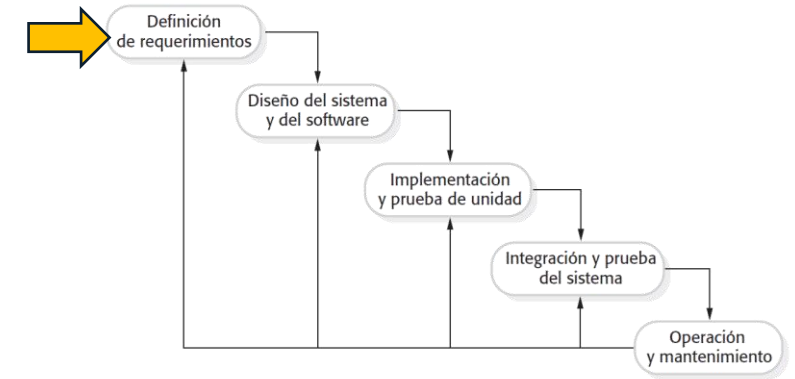
EL MODELO EN CASCADA (WATERFALL)



Smartsheet Inc. © 2022



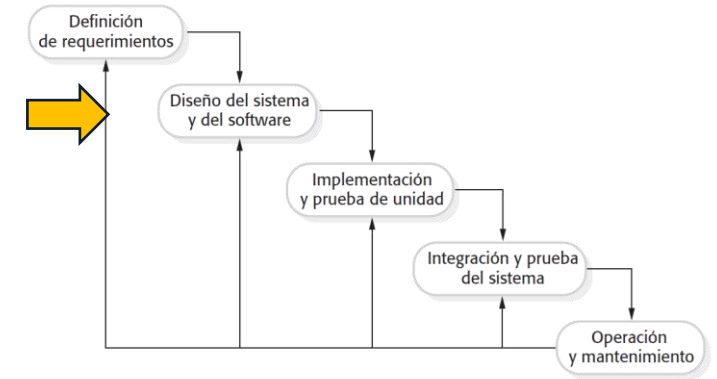
EL MODELO EN CASCADA (WATERFALL)



Análisis de requisitos del software

- En esta fase se analizan las **necesidades** de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.
- Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

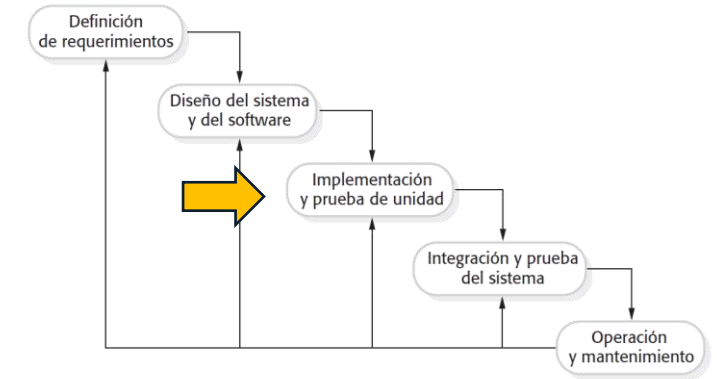
EL MODELO EN CASCADA (WATERFALL)



Diseño del sistema

- Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Descripción del diseño del software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.
- Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

EL MODELO EN CASCADA (WATERFALL)



Diseño del programa

- Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario, así como también los análisis necesarios para saber qué herramientas usar en la etapa de Codificación.

Codificación

- Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores. Dependiendo del lenguaje de programación y su versión se crean las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

EL MODELO EN CASCADA (WATERFALL)

Pruebas

- Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario, así como también los análisis necesarios para saber qué herramientas usar en la etapa de Codificación.
- Idealmente, los defectos de los componentes se detectan en las pruebas de unidad, mientras que los problemas de interfaz se localizan cuando se hacen las pruebas de integración.
- Conforme se descubren defectos el programa debe depurarse, lo cual requiera la repetición de otras etapas en el proceso de pruebas.
- El proceso es iterativo, con información retroalimentada desde etapas posteriores hasta las iniciales del proceso.

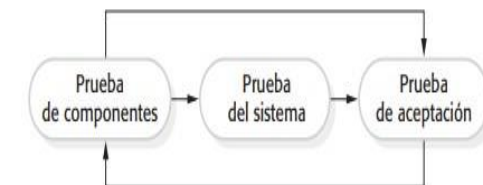
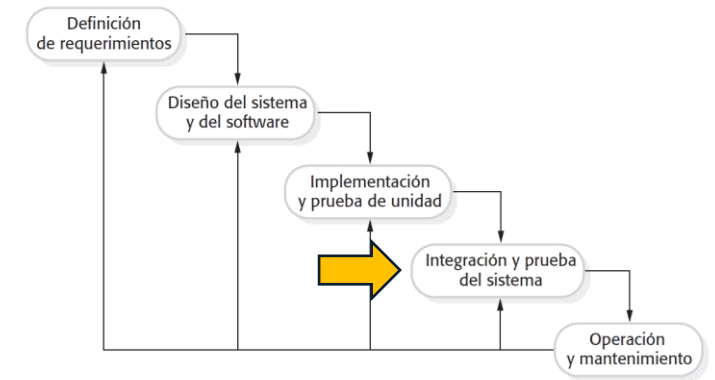
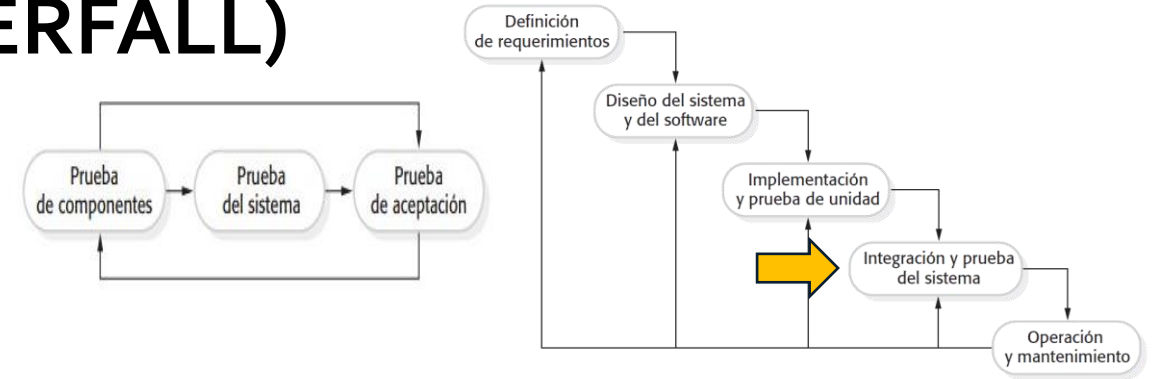


Figura 2.6 Etapas de pruebas

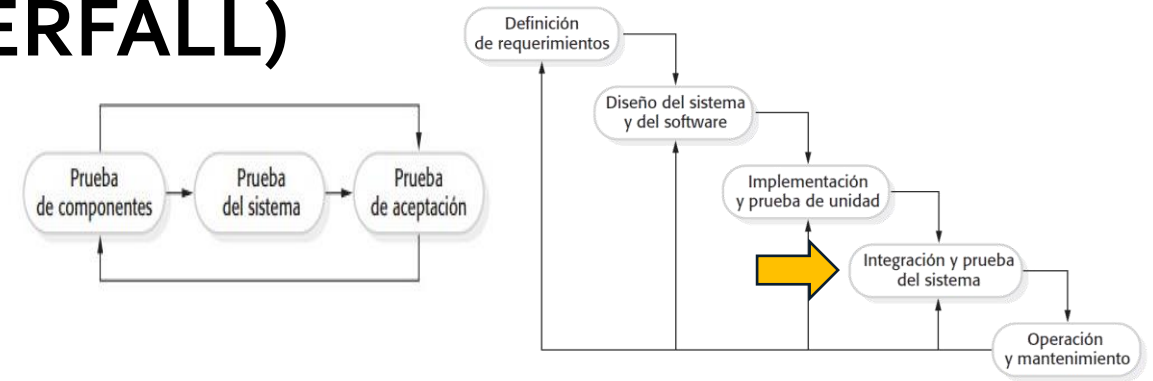
EL MODELO EN CASCADA (WATERFALL)



Pruebas del sistema:

- Aquí se prueban los componentes del sistema que se integran para crear el sistema completo.
- Este proceso tiene la finalidad de descubrir errores que resulten de interacciones no anticipadas entre componentes y problemas de interfaz de componente, así como de mostrar que el sistema cubre sus requerimientos funcionales y no funcionales, y poner a prueba las propiedades emergentes del sistema.
- Para sistemas grandes, esto puede ser un proceso de múltiples etapas, donde los componentes se conjuntan para formar subsistemas que se ponen a prueba de manera individual, antes de que dichos subsistemas se integren para establecer el sistema final.

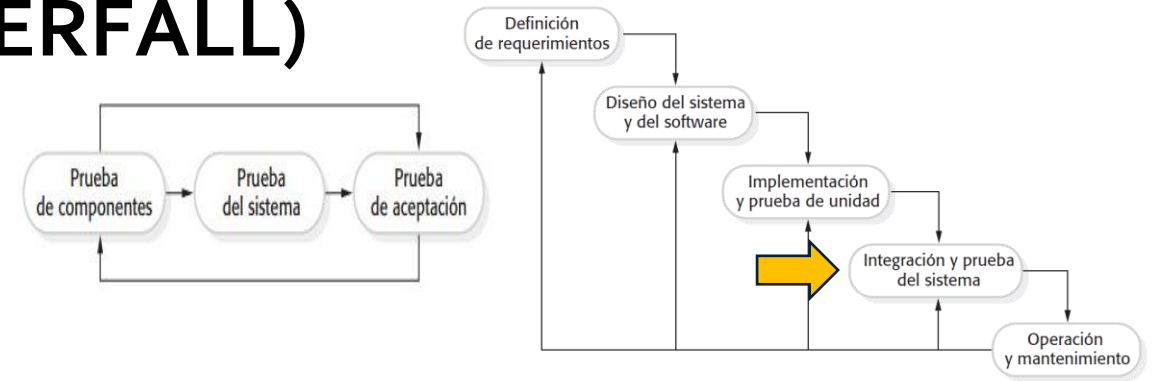
EL MODELO EN CASCADA (WATERFALL)



Pruebas de aceptación:

- El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados.
- Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba.
- Asimismo, las pruebas de aceptación revelan problemas de requerimientos, donde las instalaciones o infraestructura del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema.

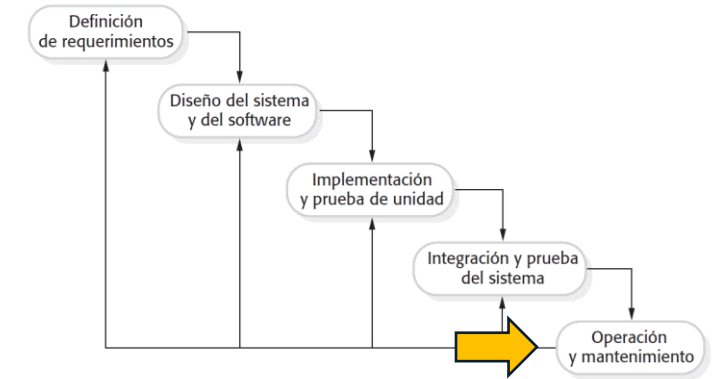
EL MODELO EN CASCADA (WATERFALL)



Pruebas de aceptación (cont.):

- A las pruebas de aceptación se las identifica como “pruebas alfa”. Se las realiza en el entorno de desarrollo y son realizadas por el cliente (o usuarios) del sistema, hasta que estén de acuerdo en que tienen una implementación aceptable de los requerimientos.
- Cuando un sistema se marca como producto de software, se realizan las denominadas “prueba beta”. Se las realiza por los usuarios del sistema en el entorno de pre-producción.
- Al encontrar errores los reportan a los desarrolladores para que los corrijan y, después de esta retroalimentación, el sistema se modifica y libera, ya sea para más pruebas beta o para su venta o entrega final.

EL MODELO EN CASCADA (WATERFALL)



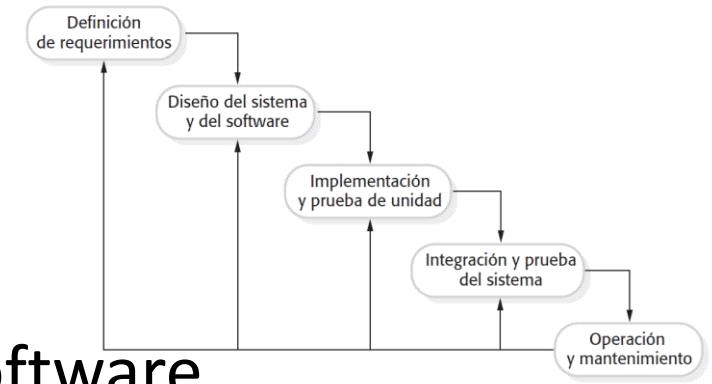
Despliegue del producto de software

- Es la fase en donde el usuario final o el cliente ejecuta el sistema, y se asegura que cubra sus necesidades, por lo que se valida o verifica el sistema.

Mantenimiento

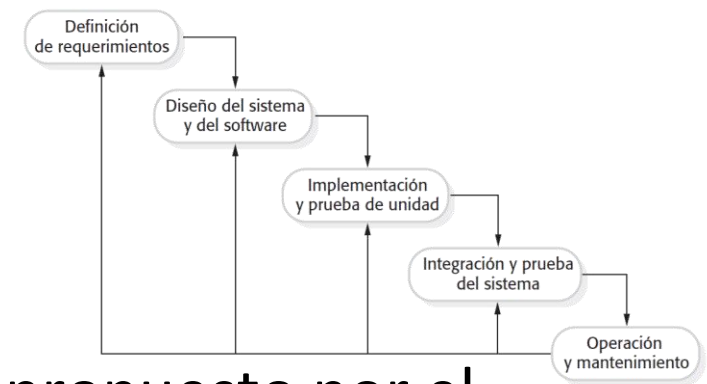
- Una de las etapas más críticas, ya que se destina un 75 % de los recursos, es el mantenimiento del software ya que al utilizarlo como usuario final puede ser que no cumpla con todas nuestras expectativas.

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 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.

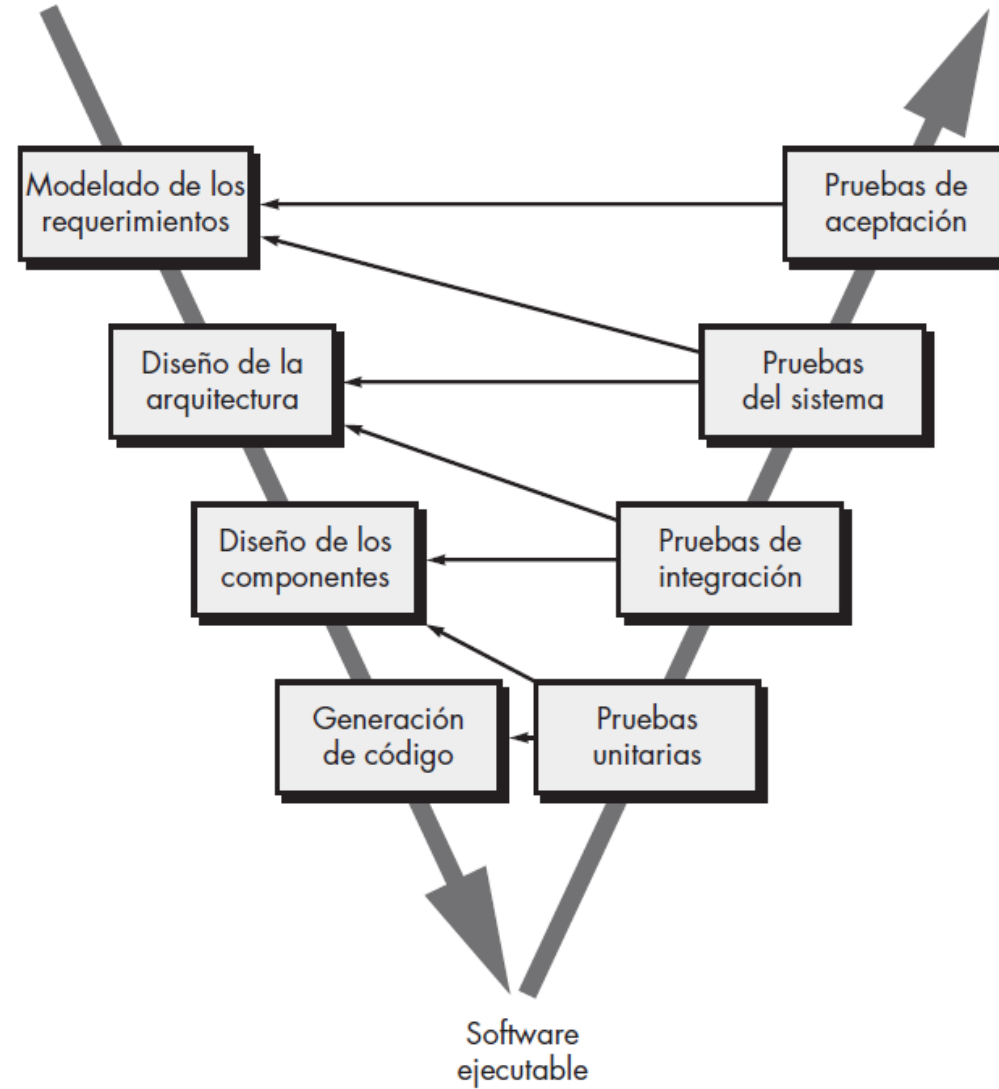
MODELOS DEL PROCESO DE SOFTWARE

- Modelo en Cascada
- ➔ • Modelo en V
- Modelo de proceso incremental
- Modelos de procesos evolutivos
 - Modelo de prototipos
 - Modelo espiral

EL MODELO EN V

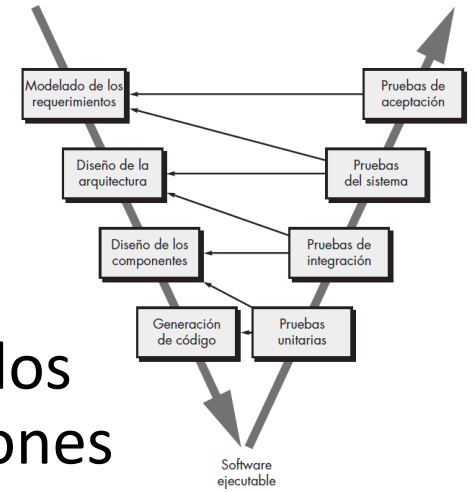
- El modelo en V

Fase

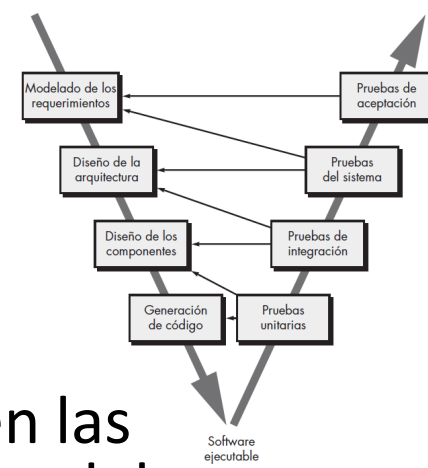


EL MODELO EN V

- 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 de 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



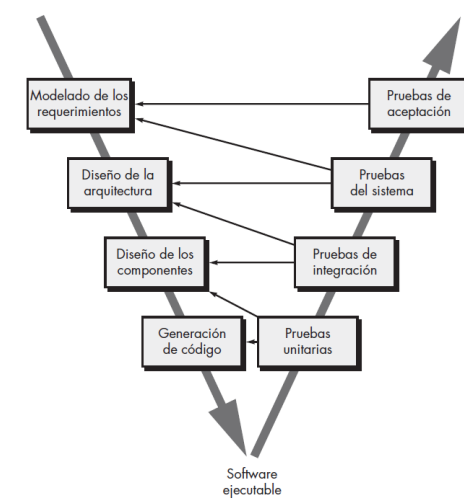
EL MODELO EN V



- **Modelado de los requerimientos:** En esta fase se identifican y definen las necesidades del cliente. Se especifica qué debe hacer el sistema, cómo debe hacerlo y en qué entorno funcionará.
- **Diseño de la arquitectura:** En esta fase se define la arquitectura general del sistema. Se identifican los componentes principales del sistema, sus interfaces y cómo se comunicarán entre sí.
- **Diseño de los componentes:** En esta fase se diseñan los componentes individuales del sistema. Se especifican los detalles de implementación de cada componente.
- **Generación de código:** En esta fase se escribe el código fuente del sistema. El código fuente se convierte en código ejecutable que puede ser ejecutado por la computadora.

EL MODELO EN V

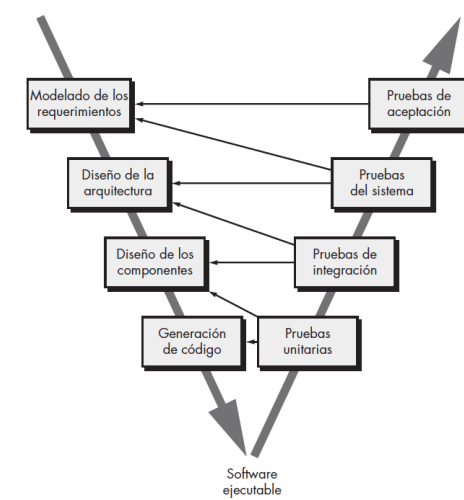
- **Pruebas unitarias:** En esta fase se prueban los componentes individuales del sistema de forma aislada. Se verifica que cada componente funcione correctamente según lo especificado.
- **Pruebas de integración:** En esta fase se prueban los componentes del sistema juntos. Se verifica que los componentes se comuniquen entre sí correctamente y que el sistema funcione como un todo.
- **Pruebas de aceptación:** En esta fase se prueba el sistema completo con el cliente. Se verifica que el sistema cumple con los requisitos del cliente. Aquí se realizan las pruebas funcionales y no funcionales (rendimiento)



EL MODELO EN V

VENTAJAS

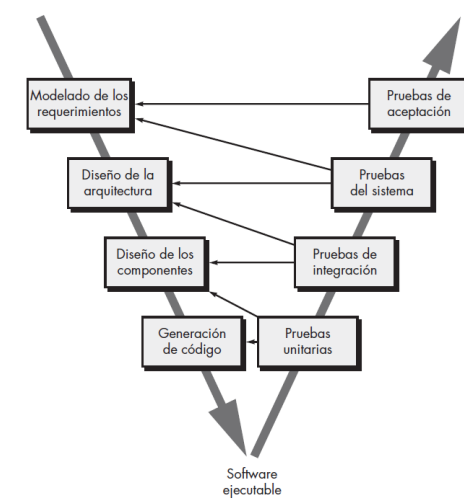
- Es un modelo simple y fácil de entender.
- Es un modelo efectivo para identificar y corregir errores en las primeras etapas del desarrollo.
- Es un modelo que fomenta la comunicación y la colaboración entre los equipos de desarrollo y pruebas.
- El modelo en V es una metodología adecuada para proyectos pequeños y medianos con requisitos bien definidos



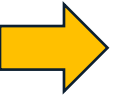
EL MODELO EN V

DESVENTAJAS

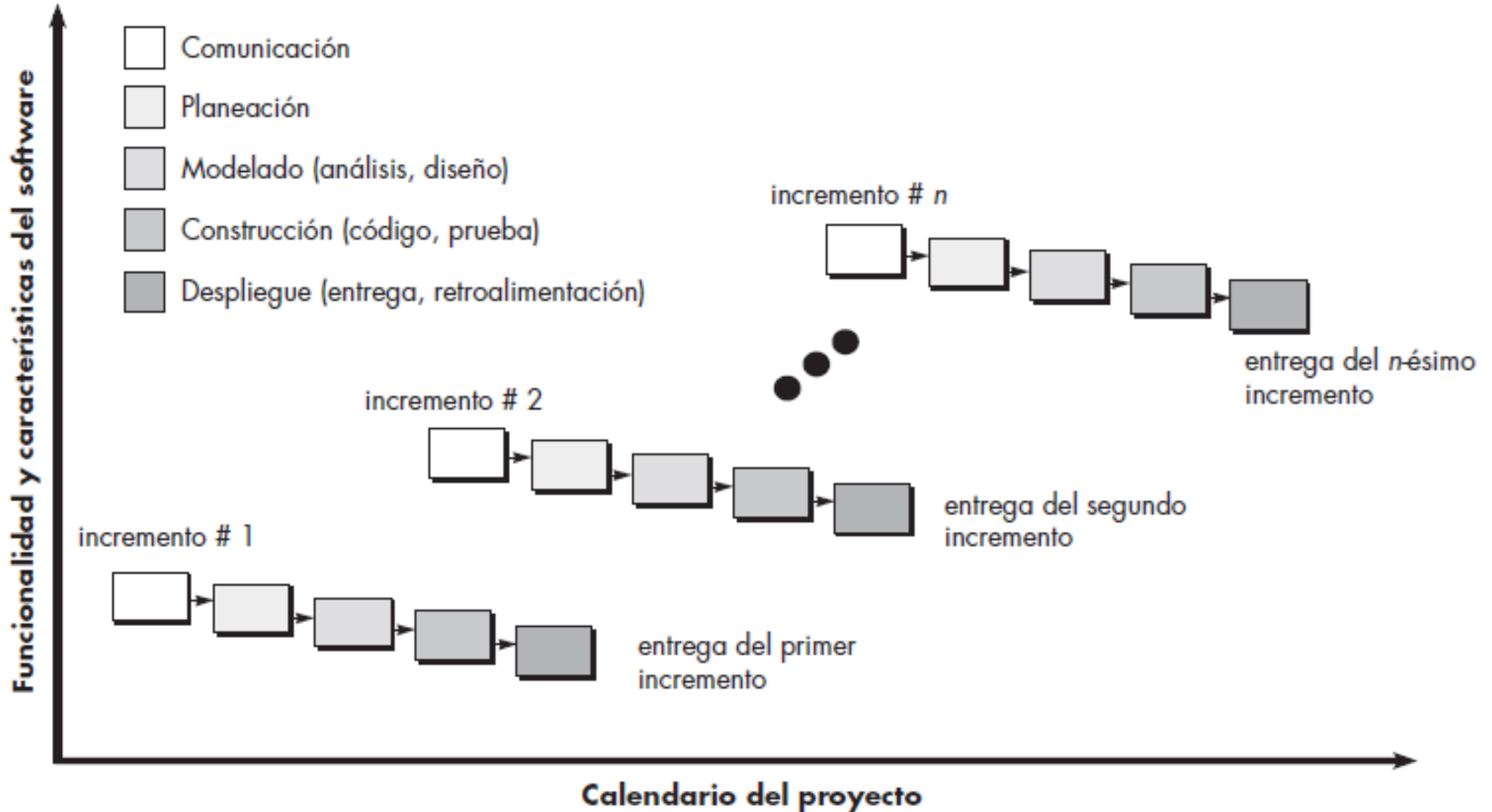
- No es flexible.
- Es difícil cambiar el diseño del sistema una vez que se ha comenzado la implementación.
- No es adecuado para proyectos grandes y complejos.
- Puede ser lento y costoso.



MODELOS DEL PROCESO DE SOFTWARE

- Modelo en Cascada
- Modelo en V
-  • Modelo de proceso incremental
- Modelos de procesos evolutivos
 - Modelo de prototipos
 - Modelo espiral

MODELO INCREMENTAL



MODELO 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 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 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 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.
- Reduce el riesgo de fracaso del proyecto. Al dividir el software en incrementos, es más fácil identificar y corregir problemas antes de que se conviertan en problemas mayores.

DESVENTAJAS:

- Con muchos incrementos se puede perder la visión del producto final.
- La integración es particularmente importante en cada incremento.

MODELO INCREMENTAL

VENTAJAS Y DESVENTAJAS

DESVENTAJAS:

- Puede ser más costoso que otros modelos de desarrollo de software. Esto se debe a que el software se entrega en una serie de incrementos, lo que requiere más trabajo de gestión de proyectos y pruebas.
- Con muchos incrementos se puede perder la visión del producto final. Puede ser más difícil controlar el alcance del proyecto. Al dividir el software en incrementos, puede ser más difícil mantener el enfoque en las funcionalidades principales del proyecto.
- Puede ser más difícil integrar los incrementos. Si los incrementos no se diseñan cuidadosamente, puede ser difícil integrarlos en un sistema completo.

MODELOS DEL PROCESO DE SOFTWARE

- Modelo en Cascada
- Modelo en V
- Modelo de proceso incremental
- • Modelos de procesos evolutivos
 - Modelo de prototipos
 - Modelo espiral

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

Entre los diferentes tipos o categorías tenemos:

- Prototipos
- Espiral

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.

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 final, 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.)

VENTAJAS:

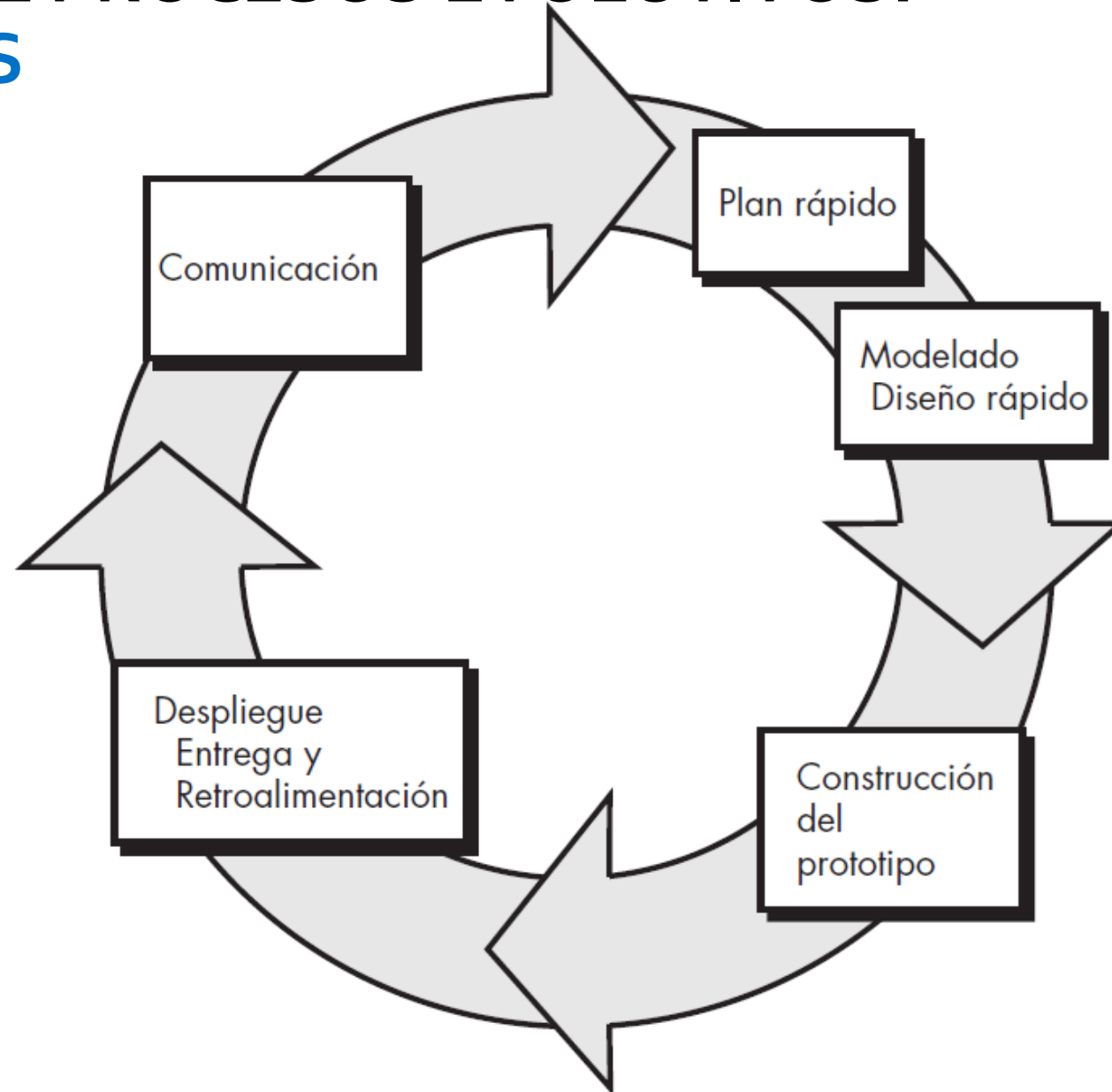
1. Participación permanente del cliente
2. Rapidez en la elaboración del producto
3. No se requieren tener todos los requerimientos desde el inicio
4. Facilidad para hacer cambios durante el proceso.

MODELOS DE PROCESOS EVOLUTIVOS: 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



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.

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 de 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:

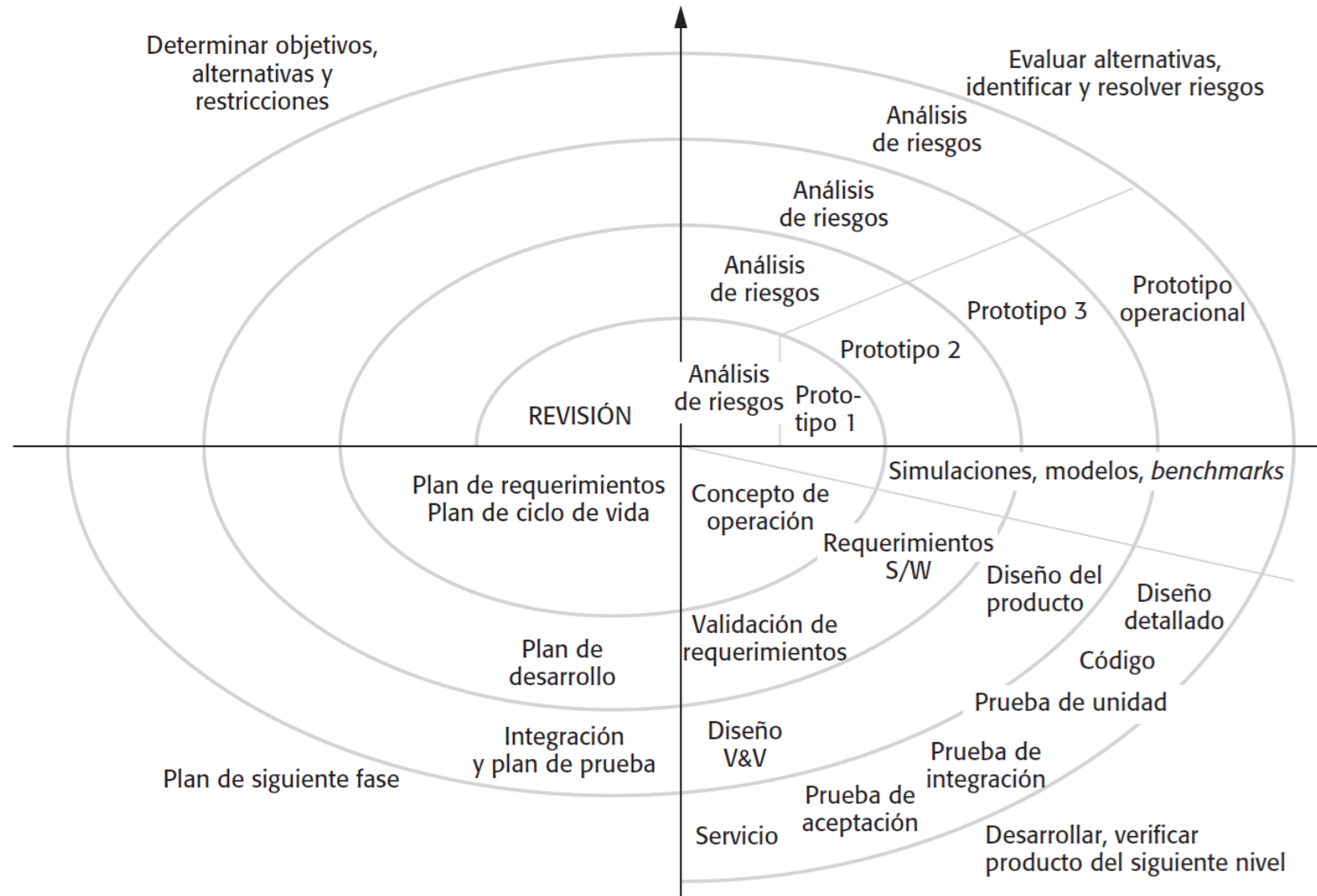
MODELO ESPIRAL (CONT.)

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:

MODELO ESPIRAL



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 final.
- 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. (Time to Market)

MODELOS DE PROCESOS EVOLUTIVOS:

CONCLUSIONES

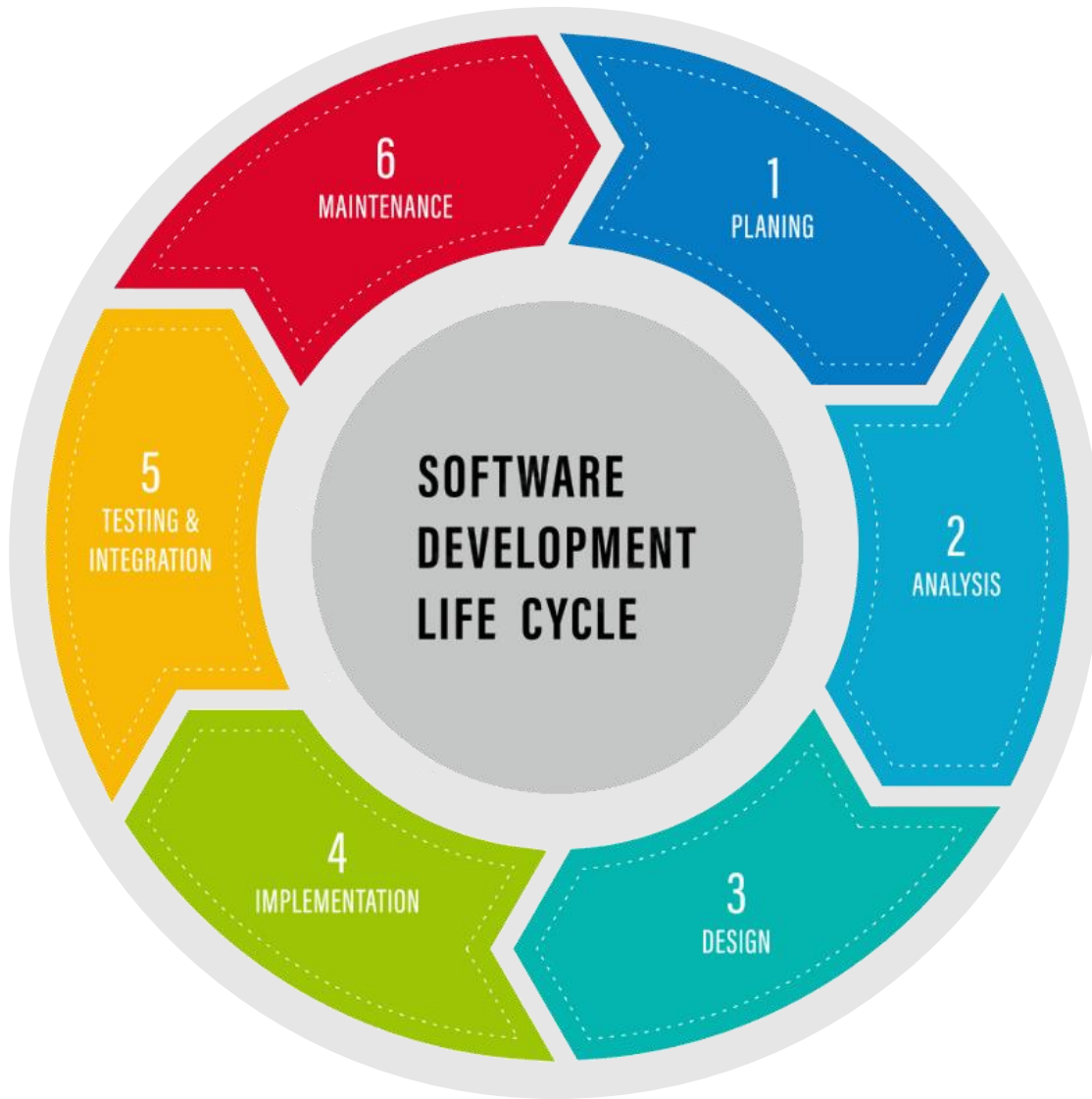
Estos modelos, sin embargo, tienen demasiadas **debilidades**:

- 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.
- 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.
- 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).



THANK YOU



VICTOR.VELEPUCHA@EPN.EDU.EC

