Ingeniería de Software
Ciclos de vida del Software

UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

ING NÉSTOR LÓPEZ LUQUE

Ciclos de vida

¿Cómo organizar en el tiempo la construcción del software?

¿En qué orden ejecutar las tareas?

¿Cuáles son las fases, hitosy entregables de un proyecto?

En el principio…

- El desarrollo de software era una tarea unipersonal
- El problema a resolver era comprendido
  - Se trataba de calcular más rápido
- El programador era el usuario de la aplicación
  - solución de ecuaciones, cálculos astronómicos, balística
- La funcionalidad era simple según estándares actuales
- El desarrollo implicaba la codificación de la solución en un lenguaje comprensible por la máquina

Codificar y corregir

- Modelo:
  - Codificar la solución
  - Corregir hasta eliminar los errores
  - Repetir mientras haya funcionalidad que agregar o cambiar
- El proceso no es planificado ni controlado
- Después de una serie de cambios:
  - La estructura del código se hace más complicada
  - Los cambios siguientes son más difíciles
  - Los resultados son menos confiables
- Es apropiado para software pequeño y bien entendido

Más que Codificar y corregir

- El desarrollo de la disciplina llevó a que:
  - Los usuarios ya no son los programadores
  - Los dominios de aplicación no siempre son conocidos por los programadores
- Además, el software debe ser:
  - Comercializado
  - Instalado en distintos ambientes
  - Probado en condiciones diferentes
- El desarrollo implica otras actividades:
  - Comprender las necesidades del cliente y los usuarios
  - Entrenar a los usuarios y asistirlos con sus dificultades

Nuevas necesidades

- Nuevos aspectos a considerar:
  - Económicos, organizacionales, psicológicos
  - La confiabilidad es esencial
    - Sistemas bancarios, control de procesos industriales, etc.
- La construcción es una actividad de equipo:
  - Estructuras de trabajo
  - Prácticas estándar
  - Planificación y control
- Claramente, construir software es mucho más que Codificar y corregir

Carencias de Codificar y corregir

- Gran complejidad del software no puede controlarse
- Cambios estructurales del software son casi imposibles
- La rotación de personal no se prevé
- La calidad no es adecuada

Consecuencia:

- Es necesario diseñar antes de codificar

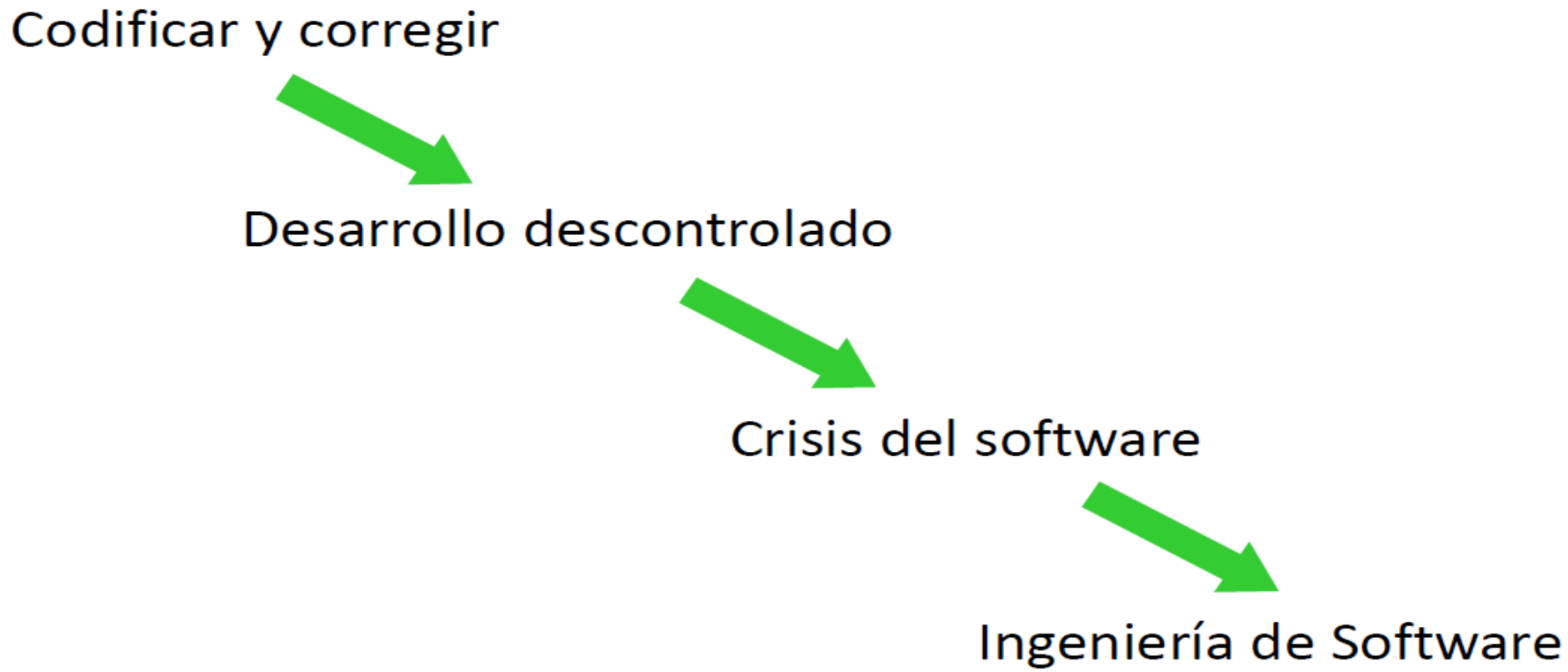
Carencias de Codificar y corregir

- El software no satisface necesidades ni expectativas del cliente
- La construcción termina fuera de plazo y presupuesto
- La flexibilidad del software es muy restringida

Consecuencia:

Es necesario analizar los requisitos antes de diseñar

Cadena de Consecuencias



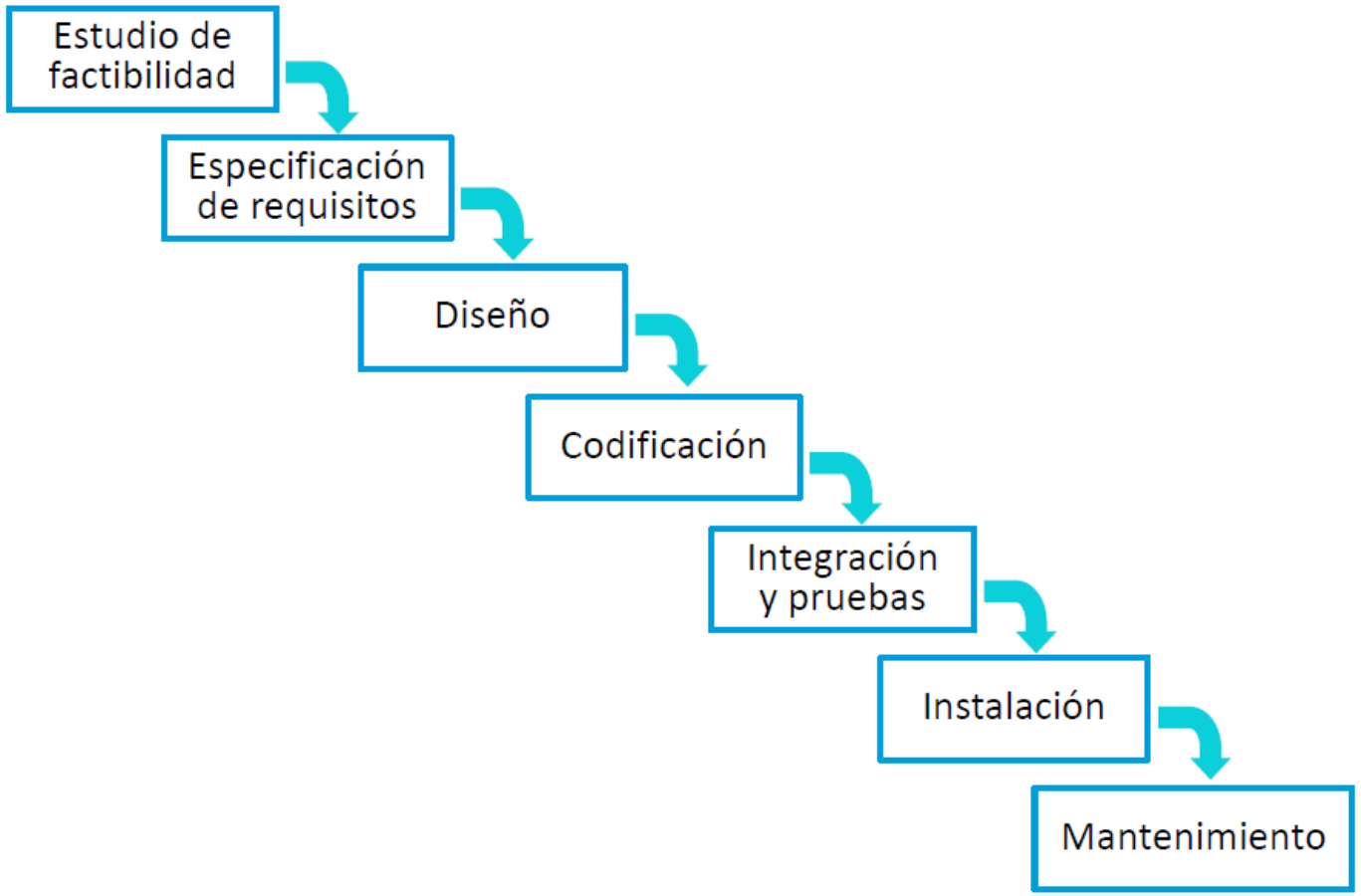
Proceso estructurado

Objetivos:

- Determinar el orden de las etapas del desarrollo y de la evolución del software
- Establecer los criterios de transición de un estado al siguiente
  - Criterios de terminación del estado actual
  - Criterios de selección de la etapa siguiente
- Un proceso estructurado determina:
  - ¿Qué es lo próximo a hacer?
  - ¿Por cuánto tiempo debe hacerse?

Cascada

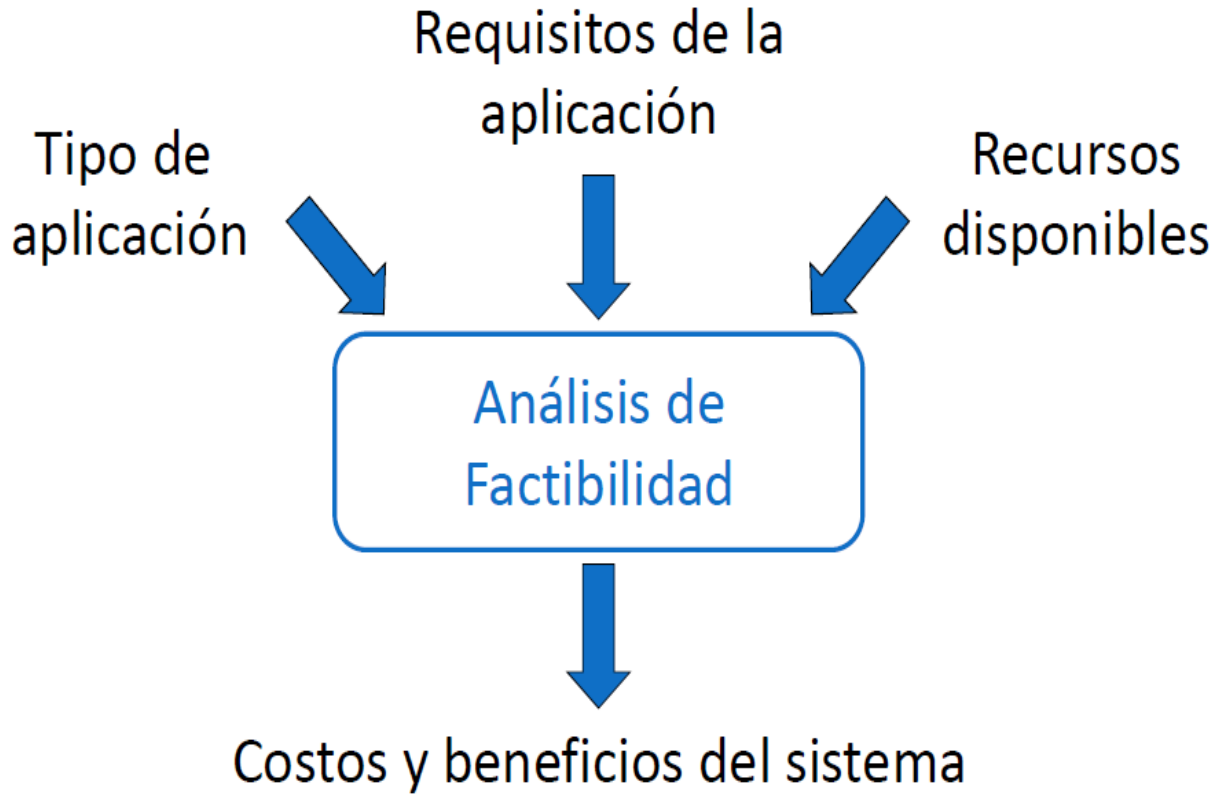
- El proceso es una “cascada” de fases
  - El producto de una fase es la entradade la siguiente
  - Es un modelo guiado por producción de documentos (document-driven)
  - Cada fase se compone de una serie de actividades que deben realizarse en paralelo
- Se popularizó en la década de los ’70
  - Guía gran parte de la práctica actual
- Existen variantes del modelo básico de cascada, pero todas comparten la misma filosofía


Estudio de factibilidad
Especificación de requisitos
Diseño
Codificación
Integración y pruebas
Instalación
Mantenimiento

Cascada

Cascada

Estudio de factibilidad



Cascada

Estudio de factibilidad:

- Se identifican posibles opciones de solución al problema planteado
- Se analizan costos y fechas de entrega por opción

Se concluye:

- Si el desarrollo vale la pena
- Bajo qué condiciones (tiempo y dinero)

Mejor opción:

- Justificación
- A veces es demasiado prematuro tomar todas las decisiones a esta altura, pero algo hay que decidir

Cascada

Mantenimiento

El análisis de requisitos es una fuente de problemas, especialmente para los usuarios finales:

- Los requisitos son difíciles de especificar
- Es muy difícil aprobar unos requisitos cuyas consecuencias no se entienden
- Los requisitos cambian con el tiempo

Algunos errores no son resueltos hasta después de instalar el software en el cliente:

- Es más caro corregir errores cuanto más tarde se detectan
- Los cambios son (casi) siempre posibles pero también (casi) siempre difíciles

Cascada

Otras actividades:

Paralelamente a las actividades esenciales del modelo de cascada, existen otras:

I. Gestión del proyecto
II. Gestión de configuración
III. Control de calidad

¿Es la documentaciónuna actividad paralela en el modelo de cascada?

Aplicación de Cascada

Se sigue una secuencia lineal de etapas

La organización decide qué artefactos deben producirse en cada etapa

- Estos artefactos permiten seguir el avance del proyecto
- Se establece qué técnicas y métodos aplicar en cada etapa
- Estas técnicas y métodos junto al ciclo de vida organizan coherentemente el “proceso de desarrollo de la organización”

Variaciones de Cascada

- Sistemas de áreas conocidas pueden omitir el estudio de factibilidad
- Sistemas complejos y/o grandes requieren dividir su desarrollo en subsistemas que pueden desarrollarse independientemente (cascadas en paralelo)
- Un sistema puede requerir una etapa de entrenamiento, transición o gestión del cambio

Análisis de Cascada

Fases claramente asociadas a las actividades de las disciplinas de la Ingeniería de Software

- Ordena el desarrollo
- Guía las actividades

Eficiente y efectivo cuando:

- ❖Las necesidades se conocen de antemano
- ❖Los métodos de diseño y desarrollo son conocidos
- ❖Se esperan pocos cambios durante el desarrollo

Ventajas de Cascada

Aplica principios de gestión al desarrollo de software

- Principios: Planificación y administración
- Esto permite predecir el desarrollo

La construcción del sistema se pospone hasta que los objetivos del sistema sean suficientemente comprendidos

- Menores riesgos tecnológicos

Desventajas de Cascada

Cascada es:

(A) Lineal: Una secuencia fija de etapas

(B) Rígido: Siempre en el mismo orden

(C) Monolítico: Todo el sistema a la vez

Nota: No se puede utilizar el sistema sino hasta después de haber llegado a fases tardías del desarrollo

Dificultad de acomodar cambios

- Pocos negocios tienen requisitos muy estables
- No apto para proyectos con innovación o incertidumbre

(A) Cascada es lineal

Un modelo lineal supone que

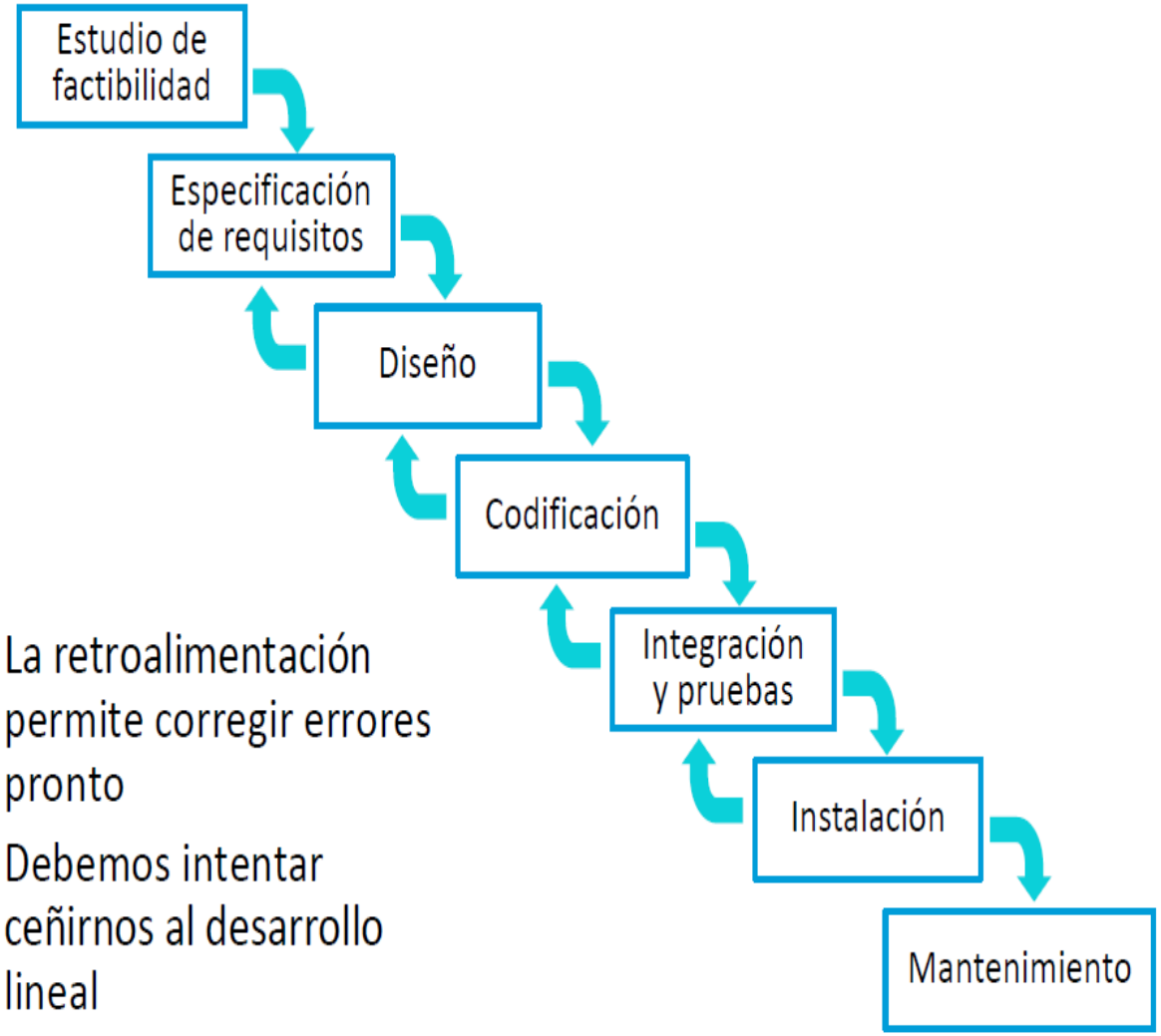
- todos los requisitos están disponibles
- el diseño elegido es el más apropiado
- la programación ocurre sin contratiempos
- los errores encontrados son fácilmente corregidos

Esto raramente sucede, ya que:

- el desarrollo comienza con requisitos incompletos
- los errores y contratiempos hacen (casi) volver a empezar

Se requiere una forma de retroalimentación

Cascada con retroalimentación



- La retroalimentación permite corregir errores pronto
- Debemos intentar ceñirnos al desarrollo lineal

(B) Cascada es rígido

Cada etapa se termina completamente antes de comenzar la siguiente

- Los requisitos están completos antes del diseño
- El cliente no participa más hasta la instalación del sistema completo

Si alguna persona termina su tarea de una etapa antes que los demás, deberá esperar a que todos terminen para comenzar la siguiente etapa

(C) Cascada es monolítico

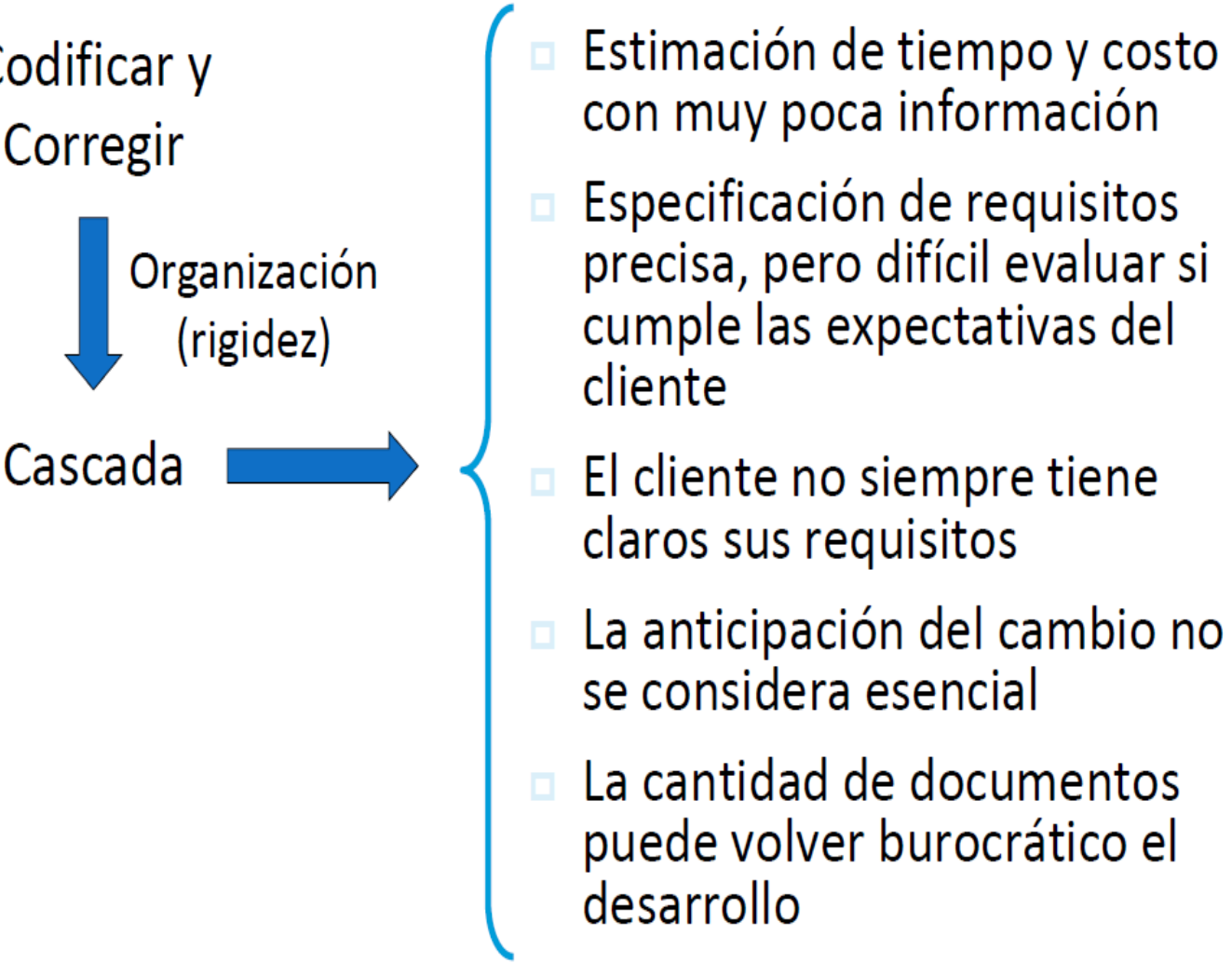
- Se planifica el ciclo de vida para entregar el sistema completo en una fecha dada
- El cliente no tiene adelantos del progreso del sistema
- Los desarrolladores construyen el sistema basados en los requisitos originales
- Los errores sólo se detectan después de instalado el sistema

Crítica a Cascada

Codificar y Corregir

Organización (rigidez)

Cascada



Do it twice

Los errores existen siempre y el software está destinado a cambiar

Do ittwice(Brooks, 1975)

- Primera versión: comprobar factibilidad y determinar requisitos críticos (prototipo desechable)
- Versión definitiva: construida con el conocimiento ganado con la primera versión (modelo de cascada)

Este modelo no soluciona la calidad monolítica de la entrega

Evolutivo

Etapas del modelo:

- Entregar al cliente algo útil
- Medir el valor agregado del incremento
- Ajustar el diseño y los objetivos en base a las mediciones

Sin rigor, el modelo evolutivo degenera rápidamente en codificar y corregir

- Para evitarlo, planificar los pasos de la evolución

Incremental

"Es el modelo cuyas etapas consisten en expandir incrementos de un producto de software operacional, donde la dirección de la evolución la dicta la experiencia con el sistema."

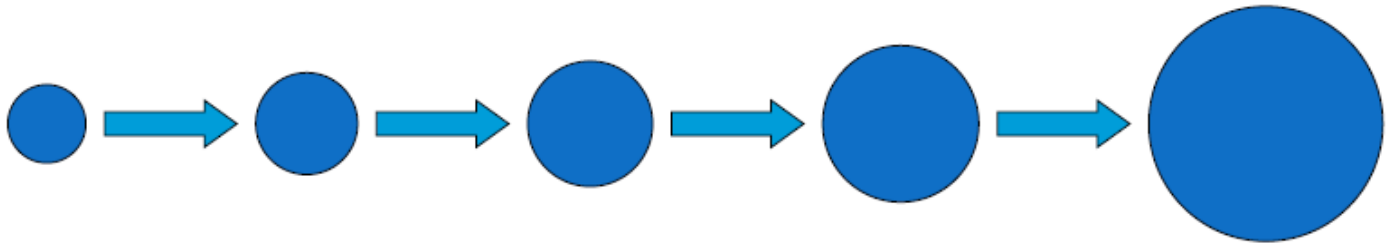
[Boehm, 1988]

Incrementos

El cliente recibe incrementos del sistema a medida que éste se va desarrollando

Entregas incrementales implican no sólo código, sino manuales, documentación, etc.

Los incrementos deben ser unidades auto-contenidas



Variaciones a Incremental

Implementación Incremental

- Se sigue el modelo de cascada hasta la etapa de diseño y luego se implementan sucesivos incrementos

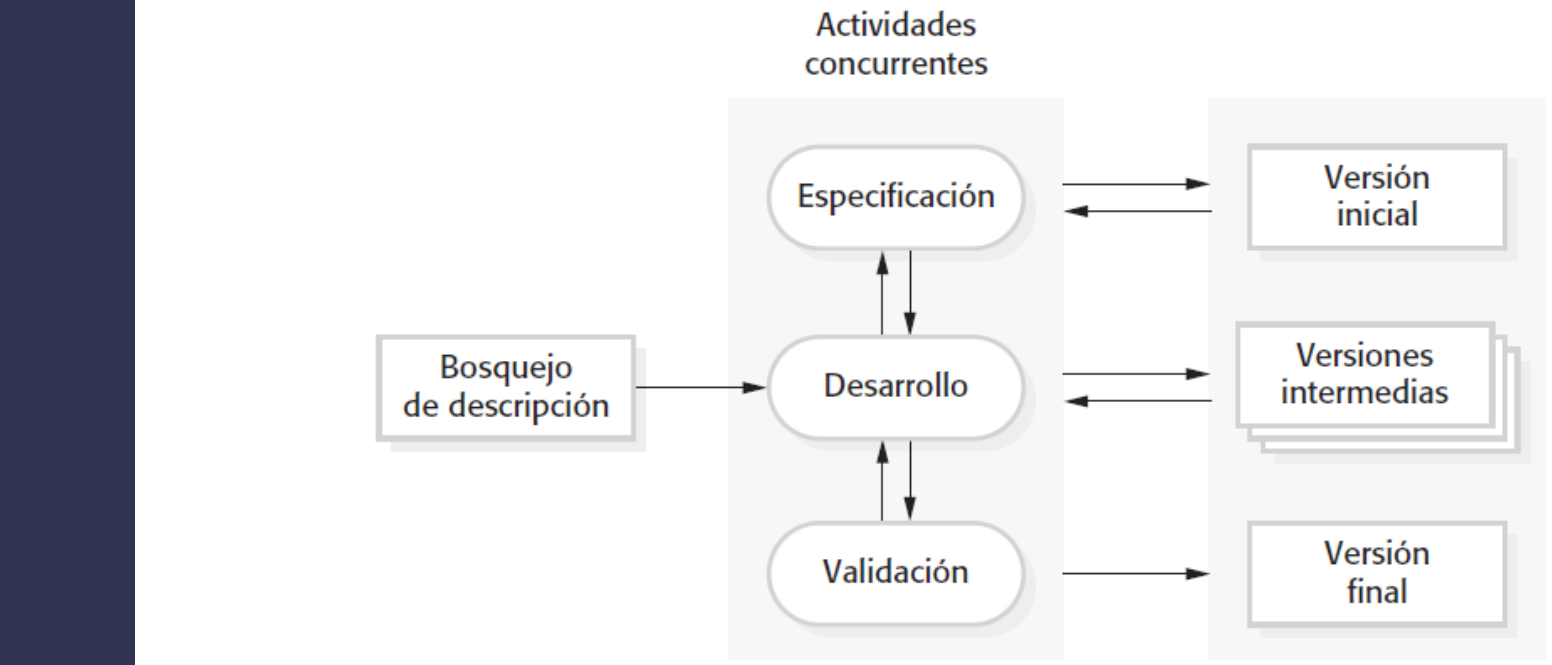
Integración Incremental

- Los módulos programados independientemente se van integrado y probando hasta tener el sistema completo

¿Diseño incremental?

¿Requisitos incrementales?

Desarrollo Incremental



Incremental

El desarrollo de software incremental, que es una parte fundamental de los enfoques ágiles, es mejor que un enfoque en cascada para la mayoría de los sistemas empresariales, de comercio electrónico y personales.

Comparado con el modelo en cascada:

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.

Desventajas

Aunque el desarrollo incremental tiene muchas ventajas, no está exento de problemas. La principal causa de la dificultad es el hecho de que las grandes organizaciones tienen procedimientos burocráticos que han evolucionado con el tiempo y pueden suscitar falta de coordinación entre dichos procedimientos y un proceso iterativo o ágil más informal.

En ocasiones, tales procedimientos se hallan ahí por buenas razones: por ejemplo, pueden existir procedimientos para garantizar que el software implementa de manera adecuada regulaciones externas (en Estados Unidos, por ejemplo, las regulaciones de contabilidad Sarbanes-Oxley). El cambio de tales procedimientos podría resultar imposible, de manera que los conflictos son inevitables.

Iterativo e Incremental

- Basado en Incremental
- El proyecto se divide en iteraciones
- Cada iteración
  - Realiza las mismas actividades
  - Construye un incremento
- Mantiene coherencia en las actividades realizadas al construir cada incremento
- En rigor, una dificultad es la carencia de una arquitectura para todo el sistema
- Orden de los incrementos guiado por riesgos

Iterativo e Incremental

Abordar el riesgo

- En cada iteración se elige el(los) requisito(s) de mayor riesgo
- Se hace un desarrollo parcial que permita evaluar con exactitud su factibilidad
- Si se logra el objetivo, el riesgo se habrá eliminado
- Si no se logra el objetivo, se aborta/redefine el proyecto antes de haber invertido más recursos

Ventajas de I&I

- ❑Los riesgos críticos son resueltos antes de hacer grandes inversiones
- ❑Las iteraciones iniciales permiten la retroalimentación de los usuarios
- ❑Pruebas e integración son continuos
- ❑Aumenta el ritmo del proyecto mediante hitos de corto plazo
- ❑La medición del avance se hace evaluando implementaciones
- ❑Se puede liberar implementaciones parciales

Comparación