



# Ingeniería de Software II (CI-4712)

---

## **Modelos de desarrollo de software**

# Referencias básicas

---

- Ingeniería de software. Un enfoque práctico. Pressman, R. Quinta edición. Mc. Graw Hill 2002
- Ingeniería de software. Sommerville, I. Séptima edición. Addison Wesley 2005



# Ingenieria de Software

---

- Un ingeniero civil supervisando la construcción de un edificio para asegurar que sea estable y de acuerdo a los requerimientos de los clientes.
- Un ingeniero o varios ingenieros escribiendo procedimientos de diversa complejidad para desarrollar sistemas de software (sw)
- El software de gran envergadura requiere de métodos, procedimientos y herramientas
- Metodologías inapropiadas incrementan costos y el consumo extra de tiempo.



# Software

---

- Características deseables
  - Confiable
  - Robusto
  - Reutilizable
  - Eficiente
  - Mantenable
  - Evolutivo
  - Portable
  - Utilizable



# Proceso de Software

---

- Proceso: Serie de operaciones usadas en la creación de un producto.
- Proceso de Software: Conjunto de tareas que tienen que ser realizadas para producir un producto de software de alta calidad (Desarrollo de software)
- Proceso de Software: Proceso que se sigue para construir un producto de software desde la concepción de una idea, hasta la entrega y el retiro final del sistema.



# Modelos de Desarrollo de Software

---

- **Actividades en el proceso de desarrollo de software**
  - Análisis de Requerimientos
  - Especificación
  - Diseño
  - Programación
  - Integración y Gestión de Configuraciones
  - Validación y Verificación
  - Prototipaje



# Modelos de Desarrollo de Software

---

- Relaciones entre las actividades: **Modelo de Desarrollo**
- Ciclo de vida del software
- Modelos de desarrollo: cascada, espiral, incremental, basado en transformaciones, basado en reutilización, etc.

# Las actividades en el proceso de desarrollo de software

---

- se relacionan según determinados:  
**modelos**
- se desarrollan aplicando un:  
**método**
- El método se fundamenta en:  
**principios**
- El método puede ser soportado por:  
**herramientas**

# Actividades usuales

---

**Identificación de requerimientos**

**Análisis**

**Programación**

**Diseño**

**Especificación**

**Validación/  
Verificación**

**Prototipaje**

**Integración**

**Gestión de configuraciones**



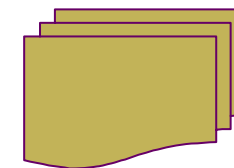
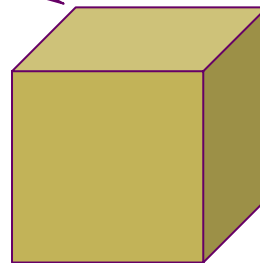
# Acerca de las actividades

---

- Se describen en forma independiente, indicando datos, rol y resultados
- Utiliza y produce documentos
- Se relacionan e interactúan de diferentes maneras conformando distintos procesos de desarrollo de software (modelos)
- De acuerdo al modelo una actividad puede jugar un rol preponderante o incluso pudiera no existir

# Análisis de requerimientos

Es una actividad  
requerida en  
cualquier  
modelo



Documentos  
orientados al  
usuario y útiles  
para el analista:

Comprensibles    Precisos  
Completos        Consistentes  
No ambiguos  
Fácil de modificar



# Análisis de requerimientos

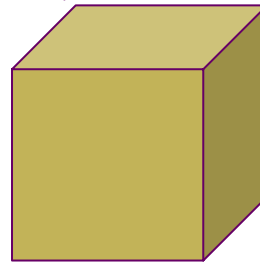
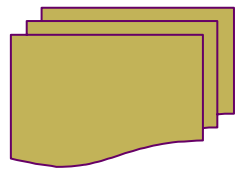
---

- Identificar el problema
- Documentar los requerimientos
- Involucrar a los usuarios y expertos en el dominio de aplicación (requiere diálogo y comunicación)
- Esta actividad puede mantenerse a lo largo del proceso
- Pueden crearse prototipos.

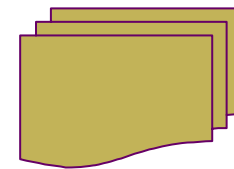
# Especificación

Describe en forma precisa el sistema a desarrollar

Consideraciones técnicas



Descripción orientada al desarrollador





# Especificación

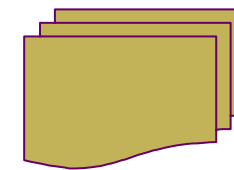
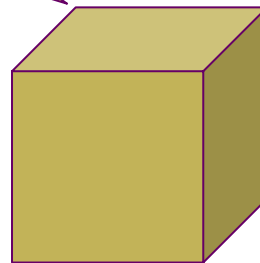
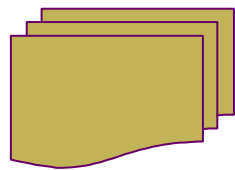
---

- Puede ser informal, semiformal o formal
- La especificación formal permite verificación
- En algunos modelos sustituye al diseño
- Describe el “qué” y no el “cómo”

# Diseño

Constituye un  
refinamiento  
del análisis

Consideraciones  
técnicas



Descripción  
detallada  
orientada al  
implementador

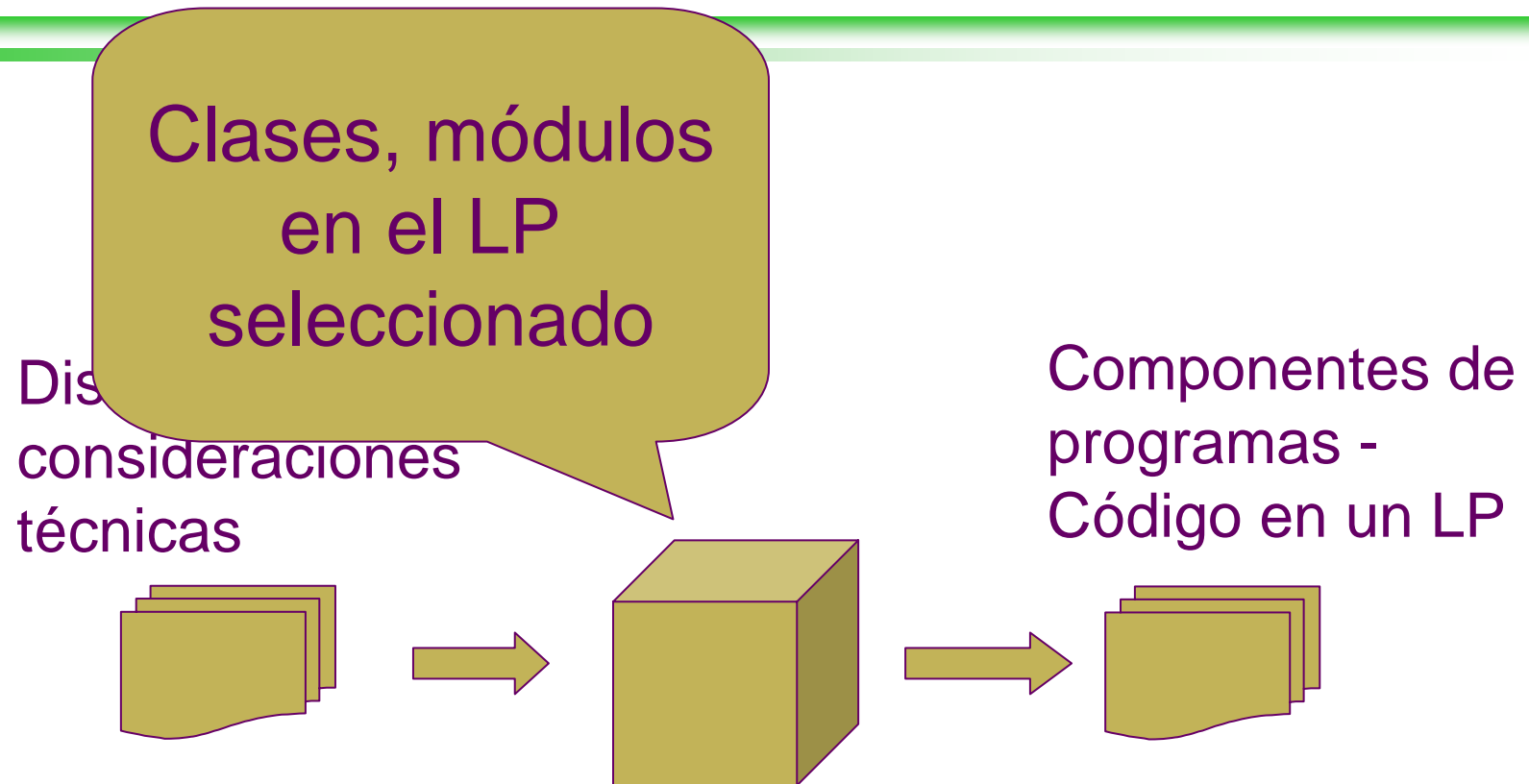


# Diseño

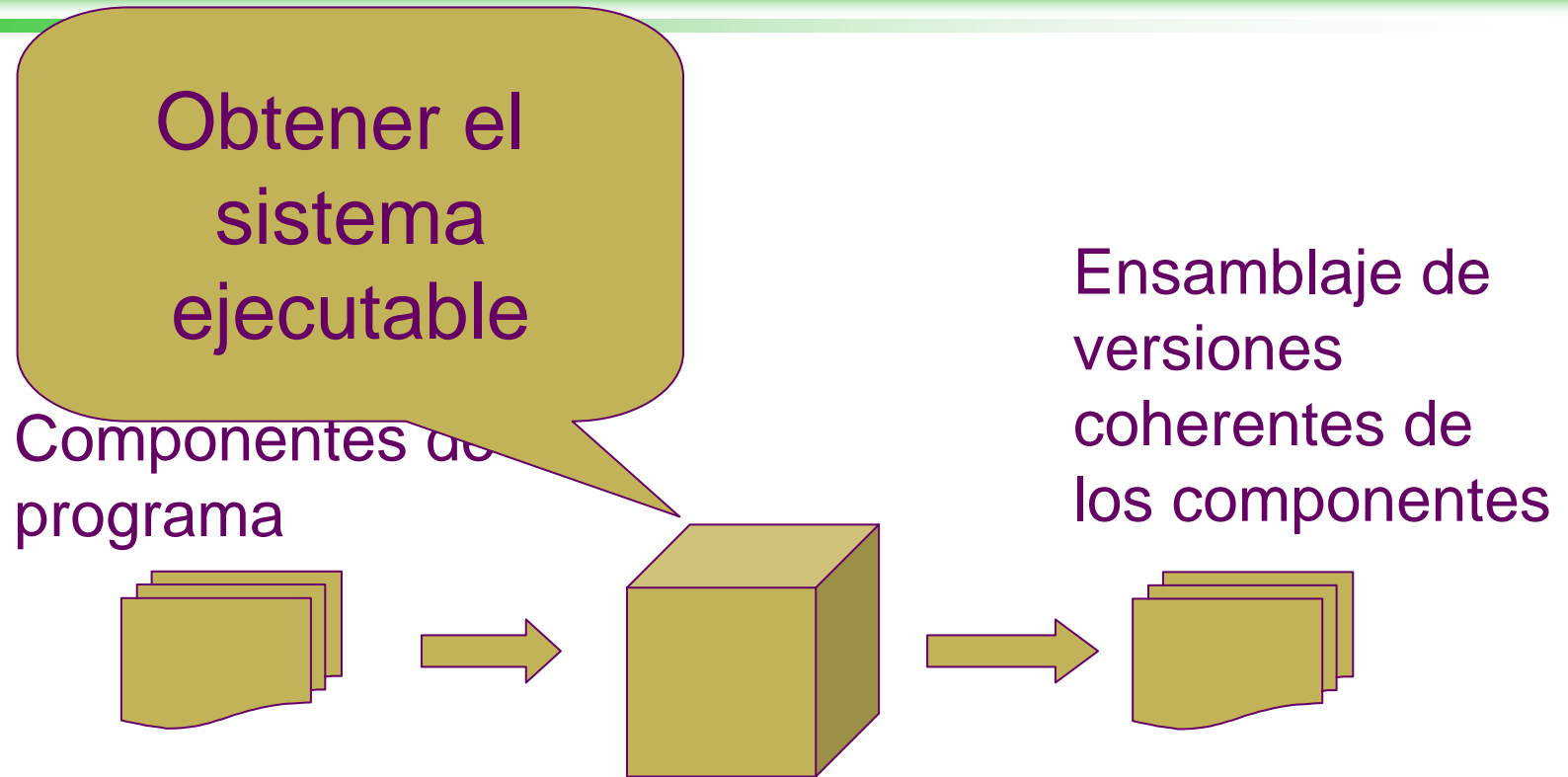
---

- Se enriquece la descripción del análisis incorporando aspectos de la plataforma de desarrollo
- Diseño arquitectónico: se desarrolla la arquitectura del sistema
- Diseño detallado: Se desarrollan los componentes (algoritmos, representación de datos, etc.)

# Programación



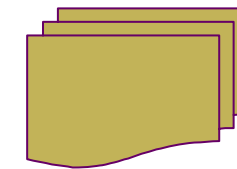
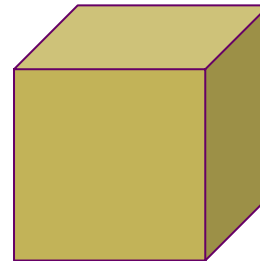
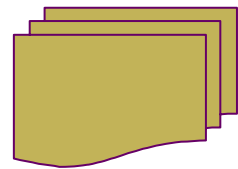
# Integración y gestión de configuraciones



# Validación y verificación

Permite  
determinar la  
confiabilidad o  
correctitud del  
producto

(textos,  
programas, etc)



Documentos  
validados/  
verificados



# Validación y verificación

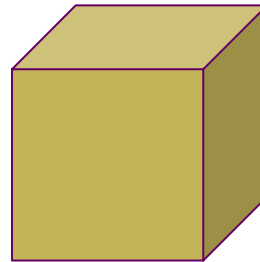
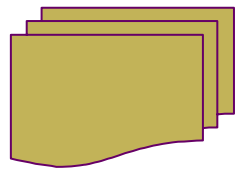
---

- Validación: el software responde a lo que espera el usuario
- Verificación: el software satisface la especificación
  - prueba: el programa satisface la especificación
  - testing: búsqueda de errores en los componentes, en la integración, en el sistema.

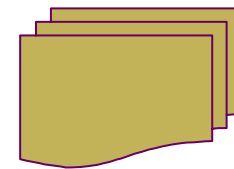
# Prototipaje

Desarrollo rápido de programas que constituyen partes del sistema

parciales del análisis



Prototipo (esbozo parcial del sistema)



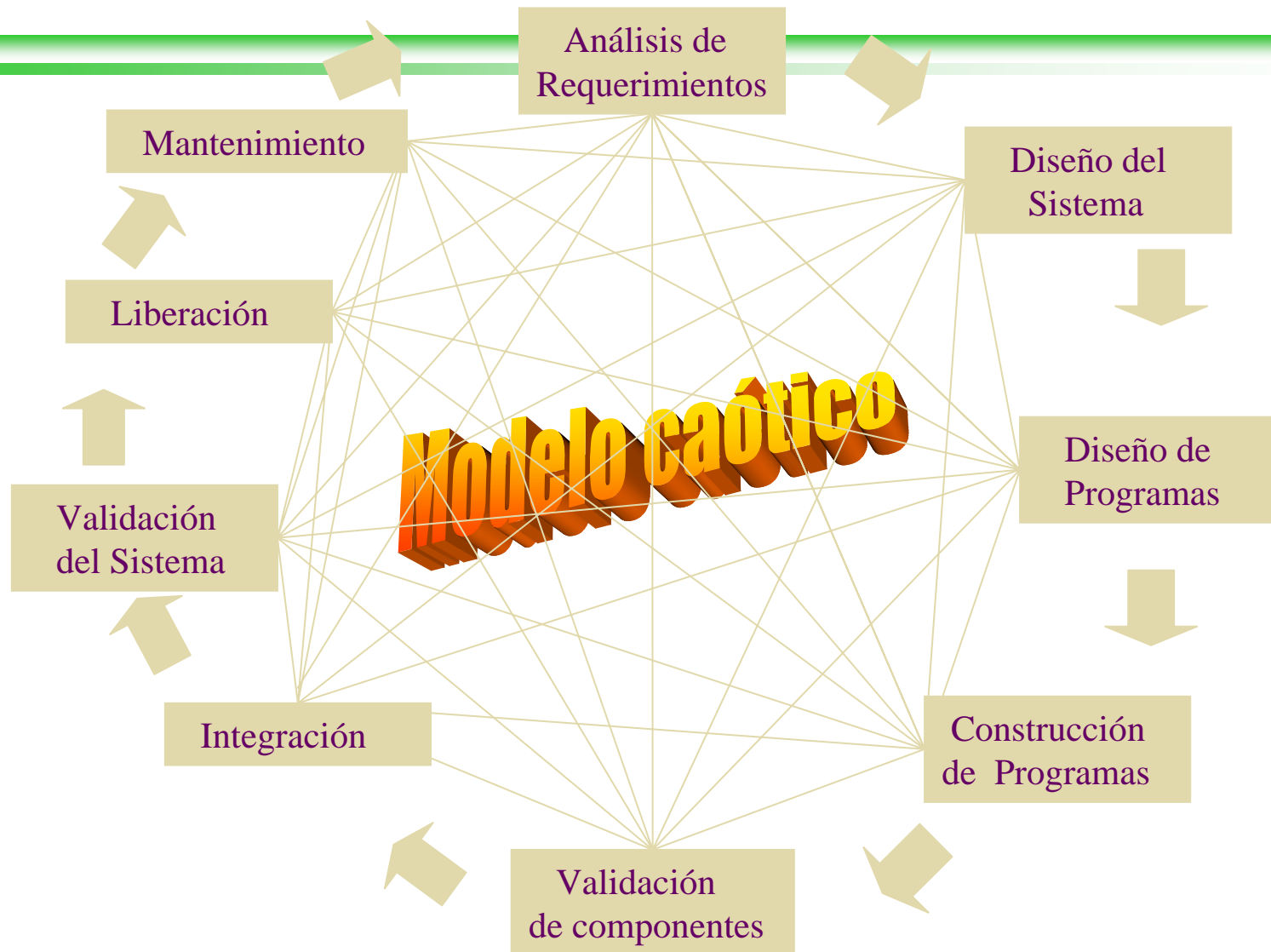


# Prototipaje

---

- Esbozo parcial del futuro sistema
- Permite experimentar
- Permite validar y precisar la especificación de requerimientos y características del futuro sistema
- Indispensable para el desarrollo de la interfaz.

# ¿Cuál modelo han usado?





# Modelos de desarrollo

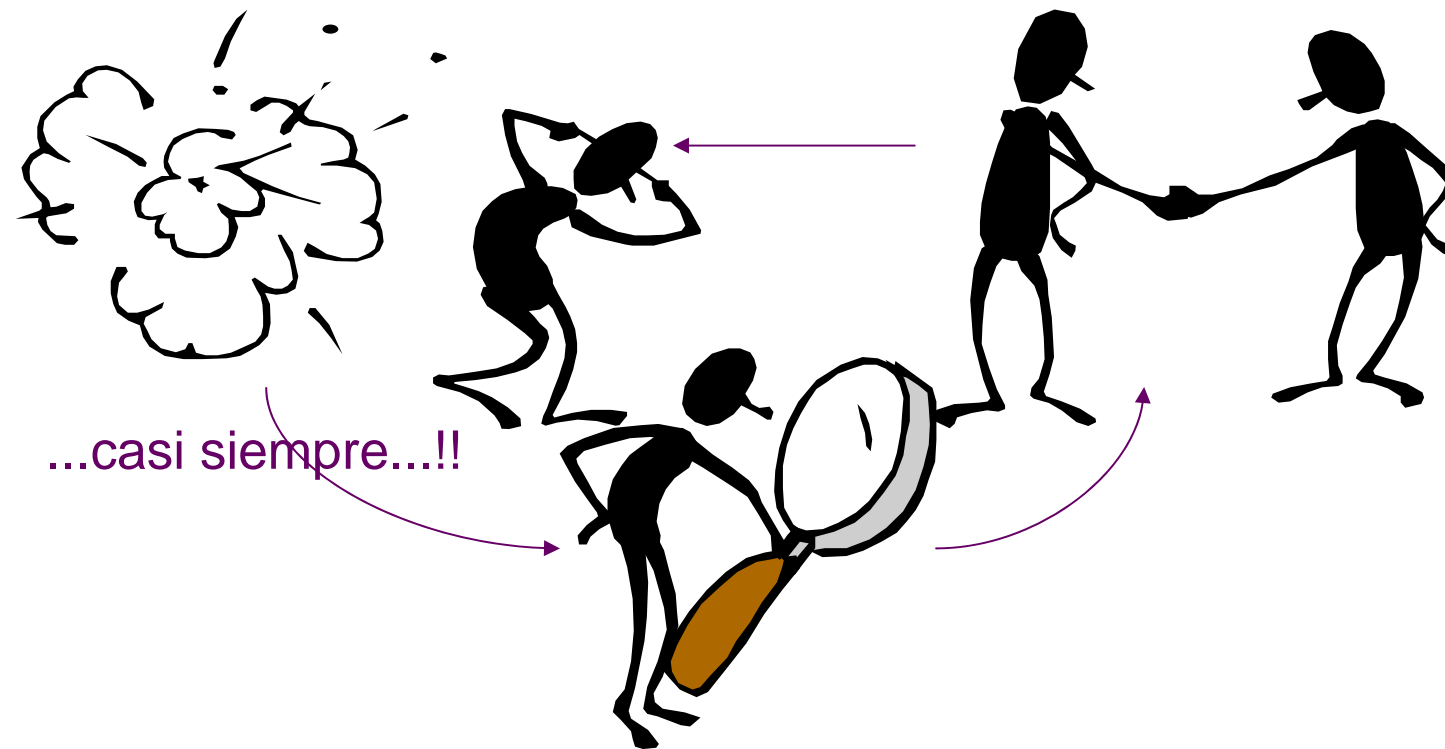
---

- Define la estructura de un proceso de desarrollo racional y controlable
- No existe un modelo universal
- Los modelos no son rígidos
- Son una guía respecto al orden en que deben adelantarse las actividades
- Se basa en el reconocimiento que el software tiene un ciclo de vida.

# Ciclo de vida del software



# Ciclo de vida del software



Mantenimiento  
del producto

# Obsolescencia del producto

---



Fin del ciclo de vida



# Modelos de desarrollo

---

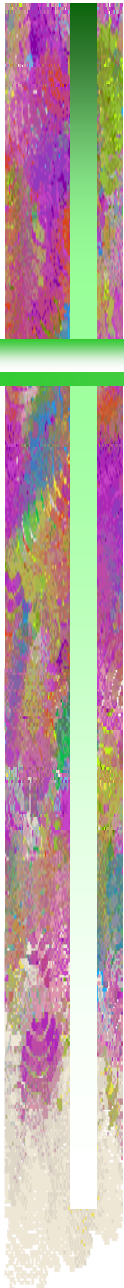
- Secuencial Lineal
  - Cascada (clásico)
  - RAD (Desarrollo Rápido de Aplicación)
- Evolutivo
  - Incremental
  - Espiral
  - Basado en reutilización
- Basado en transformaciones
- .....



# Modelo de la cascada

---

- Propuesto por Winston Royce en 1970
- Conocido como modelo secuencial lineal
- Encadenamiento secuencial de las actividades
- Cada etapa produce documentos que son la entrada a la siguiente
- Para desarrollar una etapa debe concluirse la anterior
- Popular en la década 70



# Modelo de cascada Modificado

---

- Permite retroalimentación y solapamiento entre fases.
- Es un modelo iterativo y no lineal.
- Para facilitar la terminación de metas y tareas, es normal congelar partes del desarrollo después de cierto punto en la iteración.

# Modelo de cascada

---

- Ventajas:
  - Planificación sencilla.
  - Una plantilla estructurada para ingeniería de sw.
- Desventajas:
  - Evolución de los Requisitos .
  - Resultados al final.
  - Retrasos innecesarios.
- Útil en proyectos:
  - Todas las especificaciones claras inicialmente.
  - Producto no novedoso.
  - Complejos que se entienden bien desde el principio.

# Modelo de cascada

---

- Sponga:
  - Peter y David consultan a un arquitecto para diseñar una casa. El arquitecto les presenta un documento técnico de 100 páginas. Como no entienden y para no leer el documento, ellos aceptan el diseño.
  - Un ama de casa compra cortinas por correo. Ella recibe una descripción escrita del corte y la tela.



# Modelo de Desarrollo Rápido de Aplicación - RAD

---

- RAD: Rapid Application Development
- Modelo secuencial lineal con tiempos cortos de desarrollo
- Varios equipos participando en el desarrollo
- Cada equipo maneja una parte del sistema
- Uso de herramientas de pruebas automatizadas
- En cada etapa de liberación, los productos parciales son integrados, probados y liberados

# Modelo de Desarrollo Rápido de Aplicación - RAD

---

- Desventajas:
  - Para ciclos de desarrollo extremadamente cortos: Requerimientos bien entendidos y alcance de proyecto restringido
  - Se requiere múltiples desarrolladores
  - Compromiso de desarrolladores y clientes para un tiempo de entrega corto
  - No adecuado para sistemas que no puedan ser mantenidos adecuadamente
  - No se enfoca en detalles minuciosos



# Modelo Evolutivo

---

1. Entregar algo al usuario
  2. Recibir retroalimentación del usuario
  3. Ajustar el diseño y objetivos basado a las realidades observadas
- Enfoques:
    - Incremental
    - Espiral
    - Basado en reutilización



# Modelo Incremental

---

- Desarrollo paso a paso donde las partes de algunas etapas se posponen.
- Cada etapa consiste en expandir incrementos de un producto de software operacional
- Incrementos pueden ser entregados al cliente
- Cada incremento es diseñado, codificado, probado, integrado y entregado por separado
- Los incrementos se desarrollan uno después de otro, basados en retroalimentación recibida del cliente.



# Modelo Incremental

---

- Ventajas:
  - Existe una disponibilidad limitada de recursos de desarrollo.
  - Cuando es difícil establecer todos los requerimientos por anticipado
- Desventajas:
  - Si los requerimientos crecen, la arquitectura y el diseño puede cambiar drásticamente



# Modelo en espiral

---

- Propuesto por Barry Boehm en 1988
- Desarrollo en ciclos.
- En cada ciclo:
  - se define el objetivo,
  - se analizan los riesgos,
  - desarrollo y verificación de la solución obtenida,
  - revisión de resultados y planificación del siguiente ciclo



# Modelo en espiral

---

- Se centra en algunas mejores prácticas:
  - Manejo de Riesgos
  - Orientación al Cliente
  - Desarrollo Iterativo



# Modelo en espiral

---

- Ventajas:
  - Resolución temprana de riesgos.
  - Definición de arquitectura en sus fases iniciales.
  - Basado en un proceso continuo de verificación de la calidad.
  - Ideal para productos con un nivel alto de inestabilidad de los requerimientos.



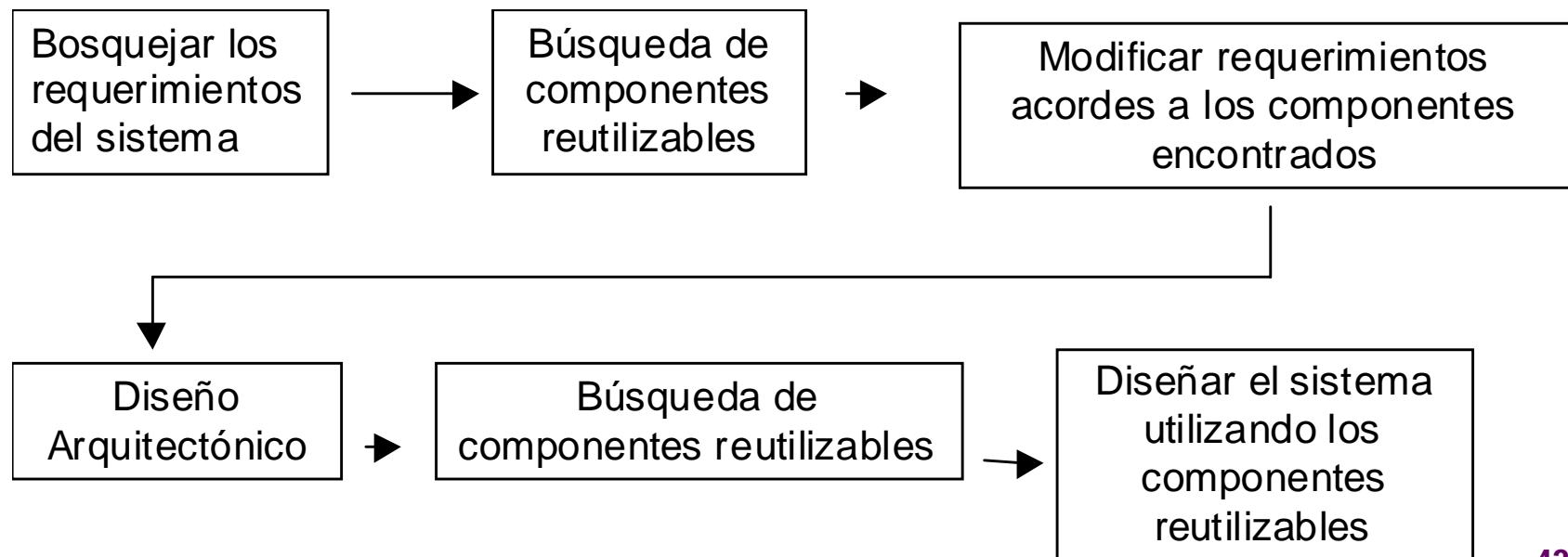
# Modelo en espiral

---

- Desventajas:
  - No aplicable a proyectos bajo contrato.
  - No recomendable en proyectos simples.

# Modelo Basado en reutilización

- Se basa en el ensamblaje de componentes





# Modelo Basado en reutilización

---

- Ventajas:
  - Incremento en la fiabilidad
  - Reducción en el riesgo
  - Utilización efectiva de especialistas
  - Conformidad con los estándares
  - Desarrollo acelerado, quizás 70% como indican algunos estudios



# Modelo Basado en reutilización

---

- Desventajas:
  - Falta de apoyo de las herramientas
  - Síndrome de aquí no se ha inventado
  - Costo de encontrar, entender y adaptar componentes reutilizables



# Modelo Basado en transformaciones

---

- Conjunto de técnicas y herramientas basadas en modelos matemáticos y lógica formal que son utilizadas para especificar y verificar los requerimientos y el diseño de sistemas computarizados.
- Se basa en especificaciones formales
- Las especificaciones son refinadas hasta alcanzar el programa
- El método formal se puede usar para verificar el sistema de una manera rigurosa usando técnicas matemáticas.

# Modelo Basado en transformaciones

---

- Una **especificación formal** es una descripción concisa del comportamiento y propiedades de un sistema escrito en un lenguaje basado en la matemática.
- Una **prueba formal** es un argumento completo y convincente para la validez de una tesis sobre la descripción de un sistema. Las pruebas formales se pueden realizar manualmente o con la asistencia de una herramienta automática.



# Modelo Basado en transformaciones

---

- Lenguajes de especificación formal: PCS (Procesos de Comunicación Secuencial), MDV (Método de Desarrollo Vienna) y la notación Z.



# Modelo Basado en transformaciones

---

- Aplicar Métodos Formales en las fases de levantamiento de requerimientos y de diseño de alto nivel.
- Las pruebas formales eliminan ambigüedad y subjetividad del análisis de los requerimientos.
- El uso de especificaciones formales y pruebas formales proveen un análisis sistemático y repetible.
- Pueden ser soportadas por herramientas de computación.



# Modelo Basado en transformaciones

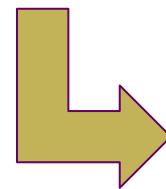
---

- Pero:
  - Es costoso
  - Consume demasiado tiempo
  - Requiere de programadores expertos en el área.

# ¿Cuál modelo?

---

- Depende del tipo de aplicación
- Pueden combinarse diferentes modelos
- Influye el contexto de desarrollo
- no existe un modelo universal



**...un proceso  
basado en funcionalidades,  
soportado en arquitecturas,  
iterativo e incremental  
UML**



# XP- Xtreme Programming

---

Se basa en la idea de ciclos o Iteraciones de desarrollo incremental

- Ciclo de Negocio
  - Release
- Ciclo de Desarrollo
  - Iteration
  - Story



# XP - Elementos Principales

---

- El Proceso de planificación
- Pequeños lanzamientos
- Metáforas: **nombres y descripciones comunes**
- Diseño simple
- Pruebas unitarias continuas
- Refactorización: **reescriben ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento**



# XP - Elementos Principales

---

- Programación por parejas
- Propiedad colectiva: **el código es modificado cuando se necesita sin retraso**
- Integración continua
- Semanas de 40 horas
- Cliente altamente disponible
- Codificación estándar



# XP - Ventajas

---

- Cambios en los objetivos y prioridades son naturales.
- Sin sobrecarga al equipo de desarrollo
- El cliente desde las primeras etapas tiene software que puede usar y probar. En cualquier momento puede parar el desarrollo, quedándose con un producto que representa lo invertido hasta esa fecha.

# XP - Desventajas

---

- Es necesario un representante del cliente en todo momento del desarrollo
- Todo el proceso de desarrollo se basa en la comunicación, si la misma es costosa o lenta perjudica enormemente el tiempo y costo del desarrollo
- No sirve para proyectos grandes debido a sus requerimientos de comunicación