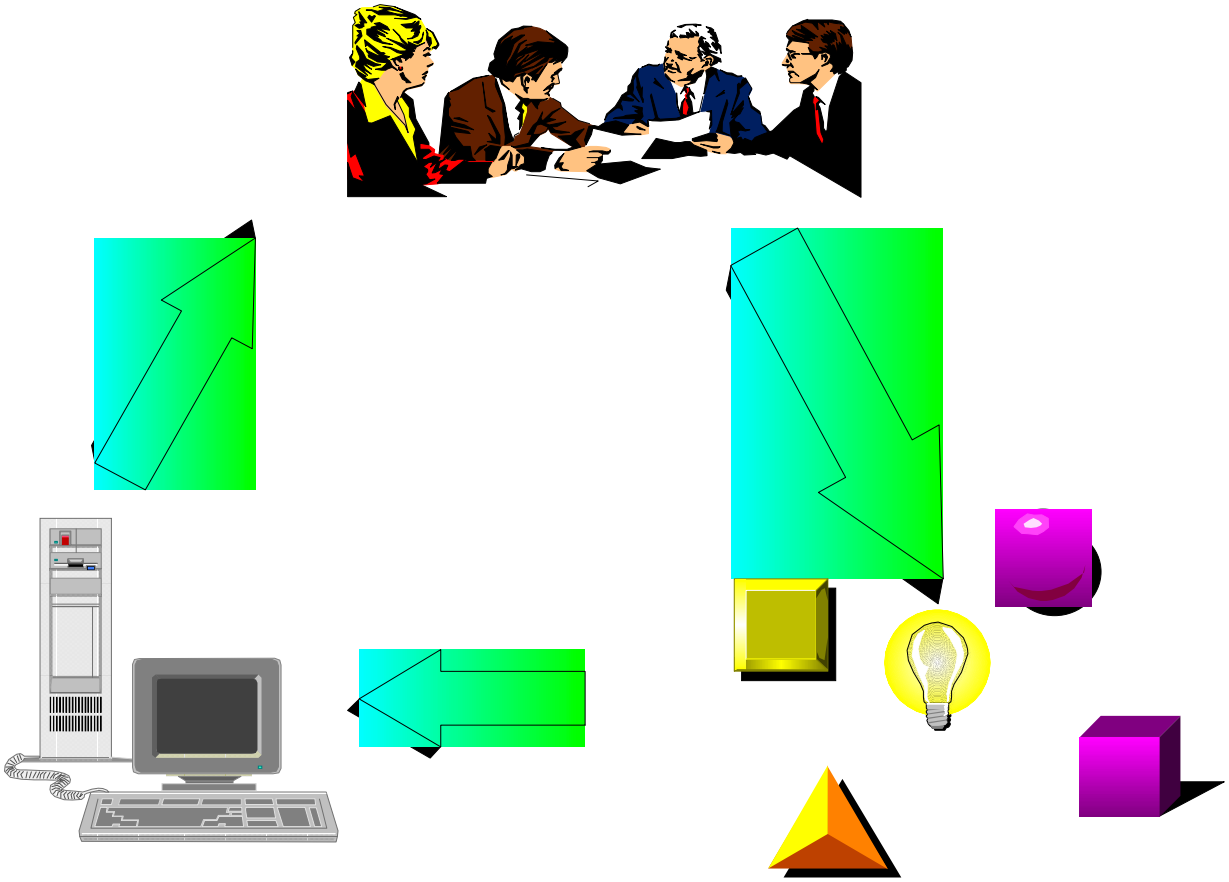
# Tema 7º
# El proceso de desarrollo

- Principios básicos
- El microproceso de desarrollo
- El macroproceso de desarrollo
- Resumen



# Principios básicos

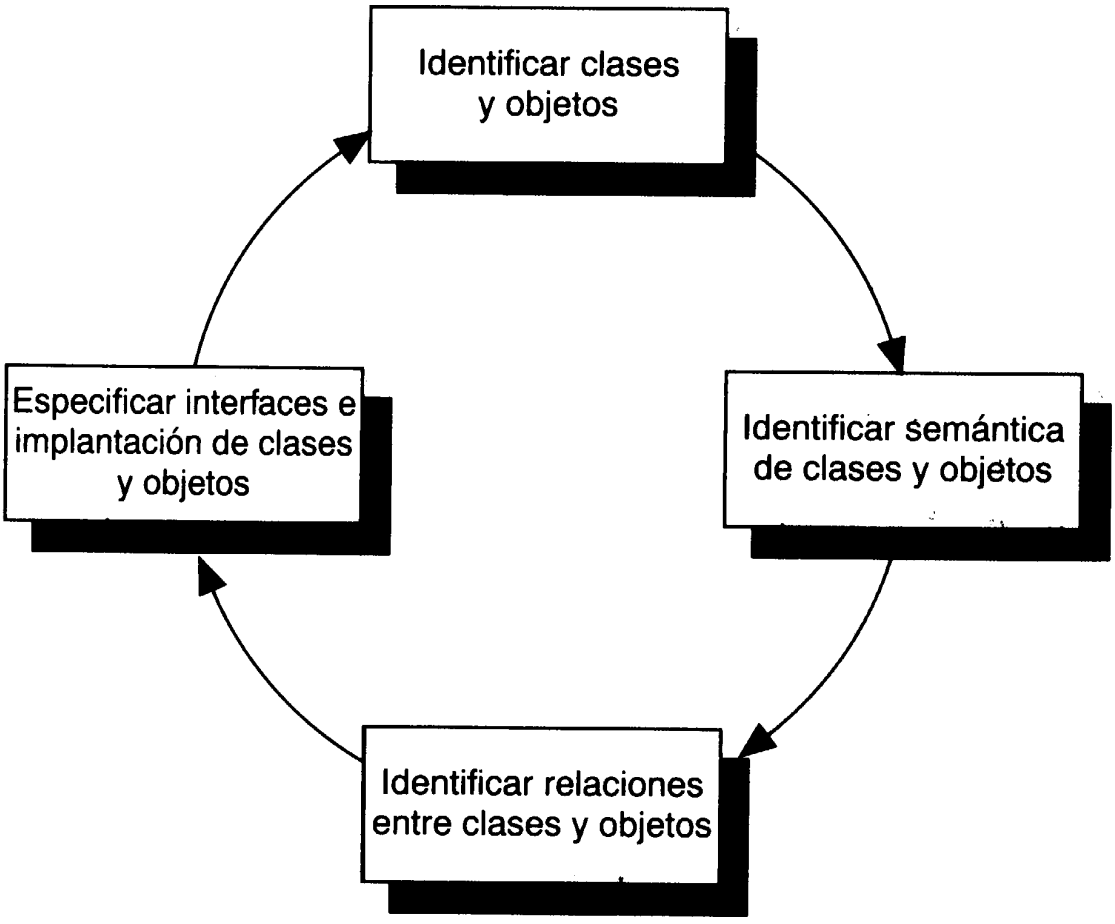
**[Booch 94]**

- No hay recetas mágicas
- Características fundamentales de un proyecto con éxito
  - **Buena visión arquitectónica**
    - No existe ningún camino bien definido para idear una arquitectura. Tan sólo se pueden definir los atributos de una buena arquitectura:
      - Capas de abstracción bien definidas
      - Clara separación de intereses entre interfaz e implementación
      - Arquitectura simple
    - Es necesario distinguir entre decisiones estratégicas y tácticas
    - ***Decisiones estratégicas*** *es aquella que tiene amplias implicaciones estratégicas e involucra así a la organización de las estructuras de la arquitectura al nivel más alto*
    - ***Decisiones tácticas*** *son las que sólo tienen implicaciones arquitectónicas locales, es decir sólo involucran a los detalles de interfaz e implementación de una clase*
  - **Ciclo de vida incremental e iterativo**
    - Los ciclos de desarrollo no deben ser anárquicos ni excesivamente rígidos
    - Cada pasada por un ciclo análisis/diseño/evolución lleva a refinar gradualmente las decisiones estratégicas y tácticas, convergiendo en última instancia hacia una solución con los requisitos reales de los usuarios finales (habitualmente no expresados explícitamente por éstos)

# El microproceso de desarrollo

**[Booch 94]**

*Representa las actividades diarias del desarrollador individual o de un equipo pequeño de desarrolladores*



# Identificación de clases y objetos

- **Propósito:** obtener las clases y objetos como abstracciones del vocabulario del dominio del problema
- **Producto:** construir el ***diccionario de datos o de clases***
  - depósito de información (repository)
    - simple base de datos
    - herramienta
- **Actividades:** Descubrimiento e invención
  - Diagramas de Casos de Uso
  - Definición de los Escenarios Principales
- **Hitos:** se completa con éxito esta fase cuando se dispone de un diccionario estable
- **Medida de la bondad:** Cuando el diccionario no sufre cambios radicales cuando se itera a través del microproceso

# Identificación de la semántica de las clases y objetos

- **Propósito:** Establecer el comportamiento y los atributos de cada abstracción identificada en la fase previa
- **Productos**
  - Se obtienen tres productos
    - **Refinamiento del diccionario de clases**
    - **Diagramas de Secuencia (UML)**
      - Diagramas de Interacción de Objetos (Booch)
      - Diagramas de Seguimiento de Sucesos (OMT)
    - **Diagramas de Colaboración (UML)**
      - Diagramas de Objetos (Booch)
  - Se pueden utilizar bases de datos o herramientas
  - La incapacidad para especificar una semántica clara es un signo de que las propias abstracciones son defectuosas
- **Actividades**
  - Se realizan tres actividades
    - Narración de sucesos (*storyboarding*): Se centra en el comportamiento no en la estructura
      - Seleccionar escenario
      - Identificar las abstracciones
      - Relatar la actividad en el escenario
      - Iterar reasignando responsabilidades
    - Diseño de clases aisladas
      - Seleccionar la abstracción y enumerar sus papeles y responsabilidades
      - Idear un conjunto de operaciones suficiente que satisfagan las responsabilidades
      - Considerar las operaciones una a una y asegurarse de que son primitivas
      - Dejar para más tarde las necesidades de construcción, copia y destrucción
      - Considerar la necesidad de completud
    - Depuración de patrones
      - En el conjunto completo de escenarios buscar patrones comunes de interacción entre abstracciones
      - En el conjunto completo de responsabilidades buscar patrones de comportamiento
      - En el conjunto de operaciones buscar patrones en la forma en que se realizan las operaciones (debe hacerse en las fases tardías del ciclo de vida)
- **Hitos**: Se completa con éxito cuando para cada abstracción se tiene un conjunto de responsabilidades y/o operaciones razonablemente suficiente, primitivo y completo
- **Medidas de bondad**: Las responsabilidades que no son simples ni claras sugieren que la abstracción dada aún no está bien definida

# Identificación entre las relaciones de clases y objetos

- **Propósito**
  - Es consolidar las fronteras y reconocer los colaboradores de cada abstracción que se identificó previamente en el microproceso
- **Productos**
  - Se producen cinco:
    - **Diagramas de Clases** (UML)
      - Diagramas de clases (Booch y OMT)
    - **Diagramas de Colaboración** (UML). Más detallados.
      - Diagramas de objetos (UML,Booch)
    - **Diagramas de Actividad** (UML)
    - **Diagramas de Estados** (UML, Booch, OMT)
    - **Diagramas de Implementación** (UML)
      - Diagramas de módulos (Booch)
      - Diagramas de Procesos (Booch)
  - En esta fase no es deseable (ni posible) expresar todos los diagramas que expresen todas las visiones concebibles. *Tan sólo se representan las más interesantes*
- **Actividades**
  - Especificación de asociaciones
    - Tomar un conjunto de clases para un nivel de abstracción
    - Considerar la dependencia dos a dos clases
    - Para cada asociación especificar el papel de cada participante, su cardinalidad y las restricciones
    - Validar las decisiones de diseño recorriendo los escenarios
  - Identificación de varias colaboraciones
    - Ilustrar la dinámica de las asociaciones de la actividad anterior en el diagrama de objetos
    - Buscar jerarquías de generalización/especialización (herencias)
    - Agrupar las clases en categorías y organizar los módulos en subsistemas
    - Capturar las decisiones anteriores en diagramas de módulos
  - Refinamiento de las asociaciones
    - Buscar nuevos patrones en las jerarquías generalización/especialización
    - Si hay patrones de estructura considerar la creación de clases que capturen la nueva estructura
    - Buscar de similar comportamiento. Hermanas en el diagrama de herencias o generalizables mediante clases parametrizadas
    - Considerar la posibilidad de simplificar las asociaciones existentes para mejorar su manejo
    - Introducir nuevos detalles tácticos: papeles, claves, cardinalidad, ...
- **Hitos**
  - Se considera finalizada esta fase cuando las abstracciones se han concretado de forma que pueden servir como planos para dirigir la implementación
- **Medidas de bondad**
  - Si las abstracciones son dificiles de implementar será sintoma de que no se ha ideado un conjunto significativo de relaciones entre ellas

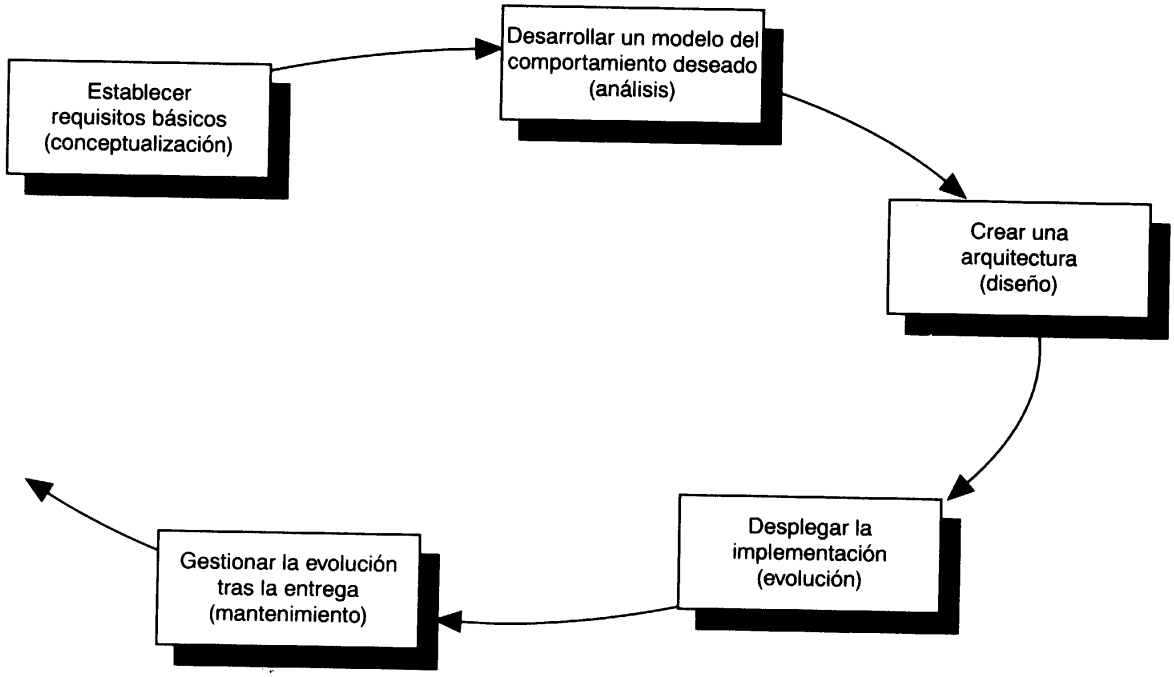
# Implementación de clases y objetos

- **Propósito:** Implementar las clases y objetos
- **Productos**
  - Un conjunto de ficheros con pseudocódigo y códigos fuente en algún lenguaje de programación
    - Se pueden generar partes con las herramientas
  - Se actualiza el diccionario de datos incluyendo la nueva información de la implementación
- **Actividades:** Selección de las estructuras de datos y algoritmos que suministran la semántica de las abstracciones identificadas previamente
  - Para cada clase identificar su protocolo con las operaciones
  - Antes de elegir una representación construida totalmente, tratar de usar la herencia
  - Procurar que los objetos no todas las operaciones y delegar en otros objetos. esto puede obligar a algún reajuste de responsabilidades
  - Tratar de usar estándares antes de crear algo propio
  - Seleccionar el algoritmo óptimo para cada operación
- **Hitos**: Se finaliza cuando se obtiene un modelo ejecutable o módulos ejecutables
- **Medidas de bondad:** Las implementaciones complejas o ineficientes indican que las abstracciones son pobres o tienen carencias

# El macroproceso de desarrollo

**[Booch 94]**

*Es el marco de referencia para controlar el microproceso*



# Conceptualización

*Establecer los requisitos principales del software que se va a construir*

- **Propósito:** *Establecer los requisitos esenciales para el sistema*
- **Productos:** *los prototipos*
  - Todos los prototipos están destinados a ser desechados
- **Actividades:** *No tiene reglas rígidas*
  - La conceptualización es una actividad creativa
  - Se pueden dar unas vías de exploración
    - Establecer objetivos
    - Reunir un equipo apropiado para hacer el prototipo
    - Evaluar el prototipo y tomar decisiones para exploraciones posteriores
- **Hitos:** *Establecer criterios explícitos para la terminación de prototipos*
- **Medidas de la bondad:** *El prototipo se considera aceptable cuando se puede dar el salto de concepto a producto*

# Análisis

## *Desarrollar un modelo del comportamiento deseado del sistema*

- **Propósito:** *Proporcionar una descripción del problema*
  - El análisis se centra en el comportamiento identificando los puntos funcionales
    - Los puntos funcionales denotan los comportamientos de un sistema observables exteriormente y comprobables
  - Para comprobar si el análisis es completo se usa la trazabilidad para comprobar que no se ha olvidado ningún punto funcional
- **Productos**
  - Casos de uso
  - Los escenarios: cada escenario denota algún punto funcional
  - Documento formal de análisis que establece los requisitos del comportamiento del sistema
  - Identificación de áreas de riesgos técnicos
- **Actividades**
  - Análisis del dominio
    - Identificar clases y objetos
  - Planificación del escenario
    - Identificar los puntos funcionales principales del sistema
    - Para cada punto funcional importante realizar una narración de sucesos
    - Generar escenarios secundarios para ilustrar comportamientos en condiciones excepcionales
    - En los ciclos de vida claves de objetos generar la maquina de estados finitos de la clase del objeto
    - Recolectar patrones entre los escenarios
    - Actualizar el diccionario de datos
- **Hitos:** *Finaliza cuando se han desarrollado y comprobado los escenarios con todos los comportamientos fundamentales del sistema*
  - La comprobación debe realizarse por los expertos en el dominio, los usuarios y los analistas
- **Medidas de la bondad**
  - Deberá comunicar una visión del completa y simple del sistema a todo el equipo de desarrollo
  - Informará a todo el equipo de desarrollo de las estimaciones de los riesgos

# Diseño

## *Crear una arquitectura para la implementación*

- **Propósito***: Crear una arquitectura para la implementación y establecer las políticas tácticas comunes que deben utilizarse por parte de los elementos dispares del sistema*
- **Productos**
  - Descripción de la arquitectura
    - Diagramas (clases, objetos,…)
    - Agrupaciones de clases y módulos
  - Descripciones de políticas tácticas comunes
    - Descripción mediante escenarios
- **Actividades**
  - Planificación arquitectónica: *Proyectar las capas y particiones del sistema*
    - Agrupación de los puntos funcionales del análisis en capas y particiones de la arquitectura
    - Validar la arquitectura creando una versión ejecutable
    - Evaluar los puntos fueres y débiles de la arquitectura
  - Diseño táctico
    - Enumerar las políticas comunes que deben seguir los elementos dispares de la arquitectura
    - Para cada política común desarrollar un escenario que describa la semántica de esa política
    - Documentar cada política y efectuar recorridos siguiéndola por toda la arquitectura
  - Planificación de versiones: Fijan el teatro donde se debe desenvolver la evolución de la arquitectura
    - Organizar los comportamientos de los escenarios siguiendo un orden (de escenarios fundamentales a secundarios)
    - Asignar puntos funcionales a cada versión
    - Ajustar objetivos y planes a cada versión, permitiendo un tiempo suficiente para documentación, pruebas y sincronización con otras actividades de desarrollo
    - Comenzar a planificar tareas y recursos necesarios para cada versión
- **Hitos**
  - Se valida la arquitectura mediante un prototipo y su revisión
  - Plan para futuras versiones
- **Medidas de la bondad:** *la simplicidad*

# Evolución

*Transformar la implementación mediante refinamientos sucesivos*

- **Propósito:** *Aumentar y cambiar la implementación mediante refinamientos sucesivos*
- **Productos:** *Conjunto de versiones sucesivas y refinadas*
- **Actividades**
  - Aplicación del microproceso
  - Gestión de cambios
- **Hitos:** *cuando la funcionalidad y calidad de las versiones son suficientes para expedir el producto*
- **Medidas de la bondad:** *Muchos cambios en los interfaces arquitectónicos y en las políticas tácticas indican mala calidad de la evolución*

# Mantenimiento

*Gestionar la evolución postventa o postentrega*

- **Propósito:** *Gestionar la evolución postventa*
- **Productos:** *Iguales a los de la fase de evolución*
- **Actividades:** *Son iguales a las de la fase de evolución más las siguientes:*
  - Asignar prioridad a las peticiones de mejoras básicas o informes de errores
  - Considerar que un conjunto amplio de cambios puede ser un nuevo punto funcional
  - Añadir pequeñas mejoras
  - Comenzar a gestionar la siguiente versión
- **Hitos:** *Nueva versión final y versiones intermedias de depuración de errores*
- **Medidas de la bondad***: flexibilidad al cambio*
  - Se entra en la etapa de **conservación** *si la respuesta a las mejoras exige recursos de desarrollo elevados*

# Resumen

- Los proyectos de desarrollo de software con éxito se caracterizan por una estructura arquitectónica fuerte y por un ciclo de vida bien dirigido, iterativo e incremental
- No hay recetas para el proceso de desarrollo
- El proceso de desarrollo orientado a objetos se puede percibir desde los puntos de vista del microproceso y del macroproceso
- El microproceso representa las actividades diarias del equipo de desarrollo
- El macroproceso es el marco de referencia que controla el microproceso y define una serie de productos medibles y actividades para gestionar el riesgo

# Referencias


- [Booch 94] G.Booch. *Object-oriented analysis and design with applications.* Benjamin Cummings (1994). Versión castellana: *Análisis y diseño orientado a objetos con aplicaciones.* 2ª Edición. Addison-Wesley/ Díaz de Santos (1996).
- [Booch 99] G. Booch, J. Rumbaugh, I. Jacobson. *The unified modeling language user guide.* Addison-Wesley (1999). Versión castellana *El lenguaje unificado de modelado.* Addison-Wesley (1999)
- [Cook 94] S. Cook and J. Daniels, *Designing Object Systems: Object-oriented Modelling with Syntropy*, Prentice-Hall Object-Oriented Series, 1994.
- [Jacobson 92] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard. *Object-Oriented software Engineering. A use case driven Approach.* Addison-Wesley (1992)
- [Jacobson 99] I. Jacobson, G. Booch, J. Rumbaugh. *The unified software development process*. Addison-Wesley (1999).
- [Piattini 96] M.G. Piattini, J.A. Calvo-Manzano, J. Cervera, L. Fernández. *Análisis y diseño detallado de aplicaciones de gestión*. RA-MA (1996)
- [Rumbaught 91] Rumbaught J., Blaha M., Premerlani W., Wddy F., Lorensen W. *Object-oriented modeling and design.* Prentice-Hall (1991). Versión castellana: *Modelado y diseño orientado a objetos. Metodología OMT*. Prentice-Hall (1996)
- [Rumbaught 99] Rumbaught J., I. Jacobson, G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley (1999)