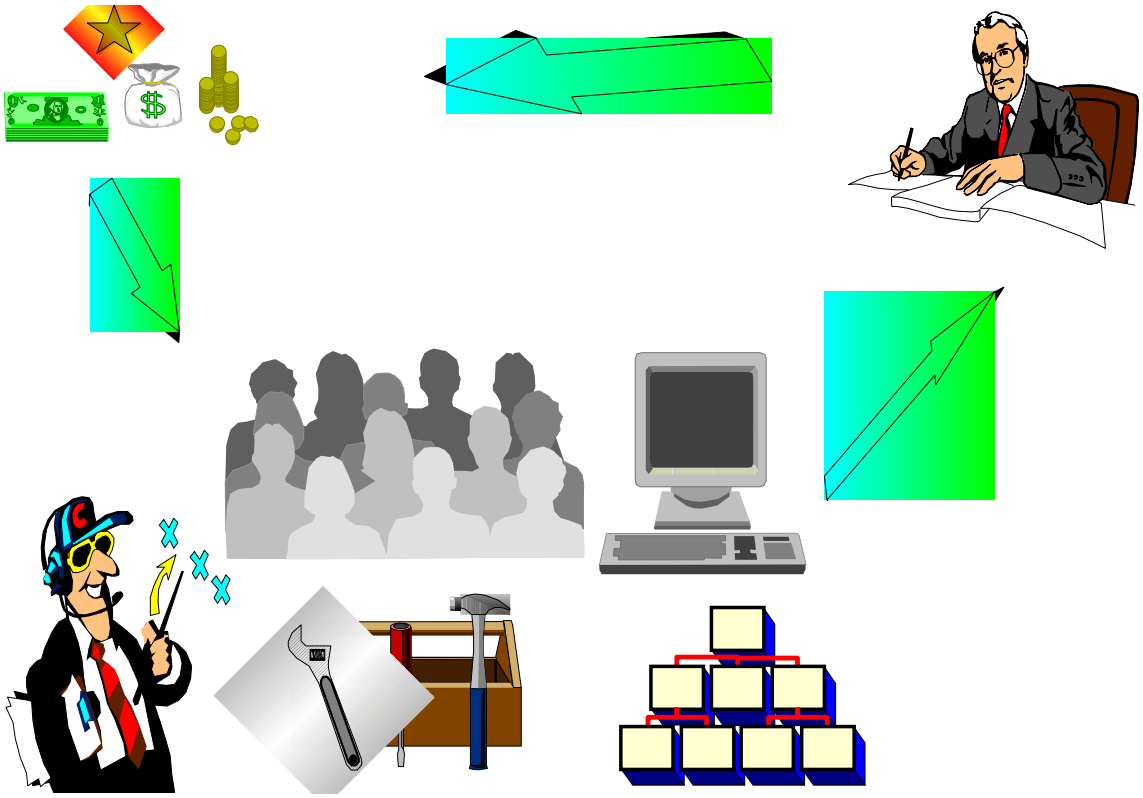
Tema 8º: Aspectos prácticos

- Gestión y planificación
- Administración de personal
- Gestión de versiones
- Reutilización
- Control de calidad del software
- Documentación
- Herramientas
- Temas especiales
- Las ventajas y los riesgos del desarrollo orientado a objetos



Gestión y planificación

Es necesario un liderazgo fuerte que gestione y dirija el proyecto de desarrollo de software

- Gestión del riesgo
  - Riesgos técnicos: *selección de las tramas de herencias, selección de herramientas,…*
  - Riesgos no técnicos: *fechas de entrega, presupuestos, relaciones con el cliente, relaciones entre los miembros del equipo,...*
- Planificación de tareas
  - Frecuencias mínimas de reuniones para favorecer la comunicaciones. Demasiadas reuniones destruyen la productividad
  - Planificar las entregas del macroproceso
  - Calibración de tiempos del equipo de desarrollo
- Recorridos de inspección
  - La dirección del proyecto debe revisar los aspectos del sistema que conlleven decisiones de desarrollo estratégicas
  - En el análisis pueden hacer revisiones incluso personal no informático para validar los escenarios planteados

Administración de personal

- Asignación de recursos
  - El primer proyecto orientado a objetos necesitará más recursos debido a la curva de aprendizaje inherente a la adopción de una nueva tecnología
  - Para el análisis los requisitos no cambian mucho respecto a otras metodologías no orientadas a objetos
- Papeles en el equipo de desarrollo
  - Papeles principales
    - Arquitecto del proyecto
    - Jefe de subsistema
    - Programador de aplicación
  - Otros papeles
    - Analista
    - Ingeniero de reutilización
    - Responsable de control de calidad
    - Jefe de integración de subsistemas
    - Documentalista
    - Responsable de herramientas
    - Administrador del sistema

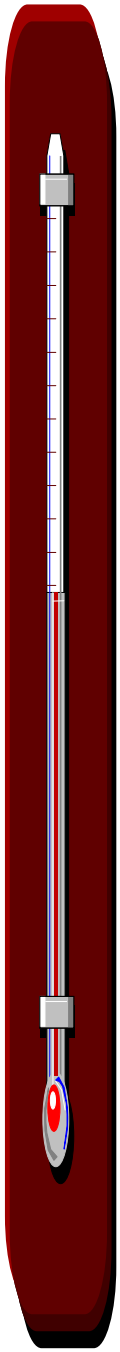
Gestión de versiones

- Integración: múltiple prototipos y versiones de producción
  - Versiones internas (pruebas alfa)
  - Versiones de prueba externas (pruebas beta)
  - Versiones definitivas
- Gestión de configuraciones y control de versiones
  - El código fuente no es el único producto del desarrollo que debería ponerse bajo la gestión de configuraciones y control de versiones. Los mismos conceptos se aplican a los productos:
    - Requisitos
    - Diagramas (clases, objetos, módulos, procesos)
    - Código fuente
    - Documentación
- Prueba
  - Prueba unitaria de clases y mecanismos
  - Prueba de subsistema
  - Prueba de sistema

Reutilización

- Elementos de la reutilización
  - Cualquier artefacto del desarrollo de software puede reutilizarse
    - código
    - diseños
    - escenarios
    - documentación
- Institucionalizar la reutilización
  - Las oportunidades para reutilizar código deben buscarse y recompensarse
  - La reutilización efectiva se alcanza mejor responsabilizando a personas concretas de esa actividad

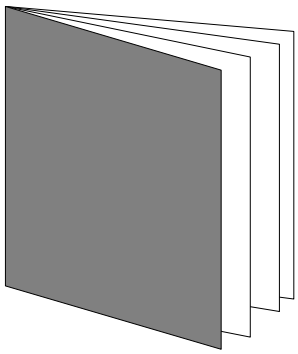
Control de calidad y métricas



- Calidad del software: *la aptitud de uso del producto completo de software*
  - El uso de tecnologías orientadas a objetos no implica automáticamente software de calidad
  - Una medida de la calidad puede ser la densidad de defectos
    - por cada mil líneas de código
    - por cada categoría de clases o por cada clase
  - La búsqueda de errores debe hacerse en las pruebas alfa y beta
- Métricas orientadas a objetos
  - ¿Cómo medir el progreso de la producción de software?
    - Líneas de código fuente (no es apropiado)
    - Horas de trabajo
    - Métrica ciclomática por clase: mide la complejidad de cada clase
    - Porcentaje finalizado de las categorías de clases y módulos
    - Métricas especiales para sistemas orientados a objetos
      - Número de métodos por clase
      - Profundidad del árbol de herencias
      - Número de hijos por árbol
      - Acoplamiento entre objetos (conexión entre objetos)

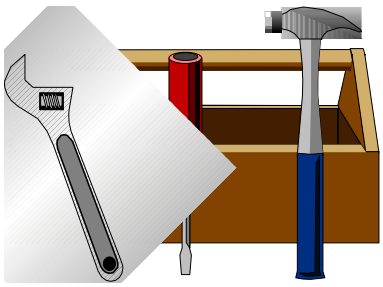
Documentación

- **El legado del desarrollo**: El propio desarrollo tambien deja una documentación que refleja:
  - Conocimiento sobre el progreso del proyecto
  - Legado de decisiones de análisis y diseño para posibles mantenedores del sistema
- **Contenidos de la documentación**
  - Documentación de la arquitectura de alto nivel del sistema
  - Documentación de las abstracciones y mecanismos clave de la arquitectura
  - Documentación de los escenarios que ilustran el comportamiento práctico de aspectos clave del sistema
- No interesa documentacion que describa clase por clase sin explicar sus relaciones



Herramientas

- Tipos de herramientas
  - Elegir herramientas que soporten el cambio de escala
  - Siete tipos de herramientas
    - Herramienta gráfica que soporte la notación
    - Browser (inspector de objetos u hojeador)
    - Compiladores incrementales
    - Depuradores que reconozcan la semántica de clases y objetos
    - Analizadores de rendimiento (profilers)
    - Herramientas de gestión de configuraciones y control de versiones
    - Bibliotecario de clases
    - Constructor de interfaces de usuario
- Implicaciones en la organización
  - Debe haber
    - un responsable de herramientas
    - un ingeniero de reutilización (mantiene la biblioteca de clases)



Temas especiales

- Cuestiones específicas del dominio
  - Diseño de los interfaces de usuario
  - Manejo de bases de datos. Se pueden encapsular como utilidades de clase.
  - Manejo de sistemas en tiempo real
  - Sistemas legados *son aplicaciones para las que existe una gran inversión de capital en software que no pueden abandonarse por razones de economía o seguridad.* Sin embargo sus costes de mantenimiento pueden ser intolerables. Se pueden encapsular como si fueran utilidades de clase.
- Transferencia tecnológica
  - ¿Cómo formar a los desarrolladores orientados a objetos?
  - ¿Qué pendiente tiene la curva de aprendizaje?

Las ventajas y los riesgos del desarrollo orientado a objetos

- Las ventajas del desarrollo orientado a objetos
  - Competitividad
  - Mayor flexibilidad (reutilización y uso de componentes)
  - Calidad
- Los riesgos del desarrollo orientado a objetos
  - Eficacia
  - Costes de puesta en marcha de una nueva tecnología

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)