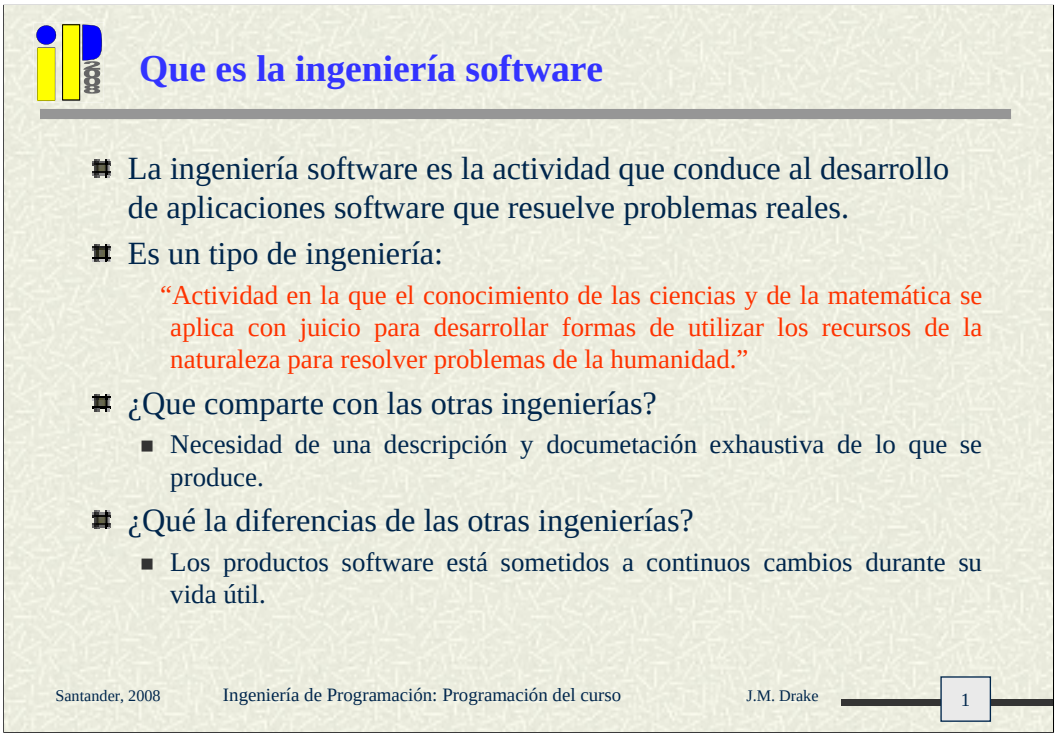
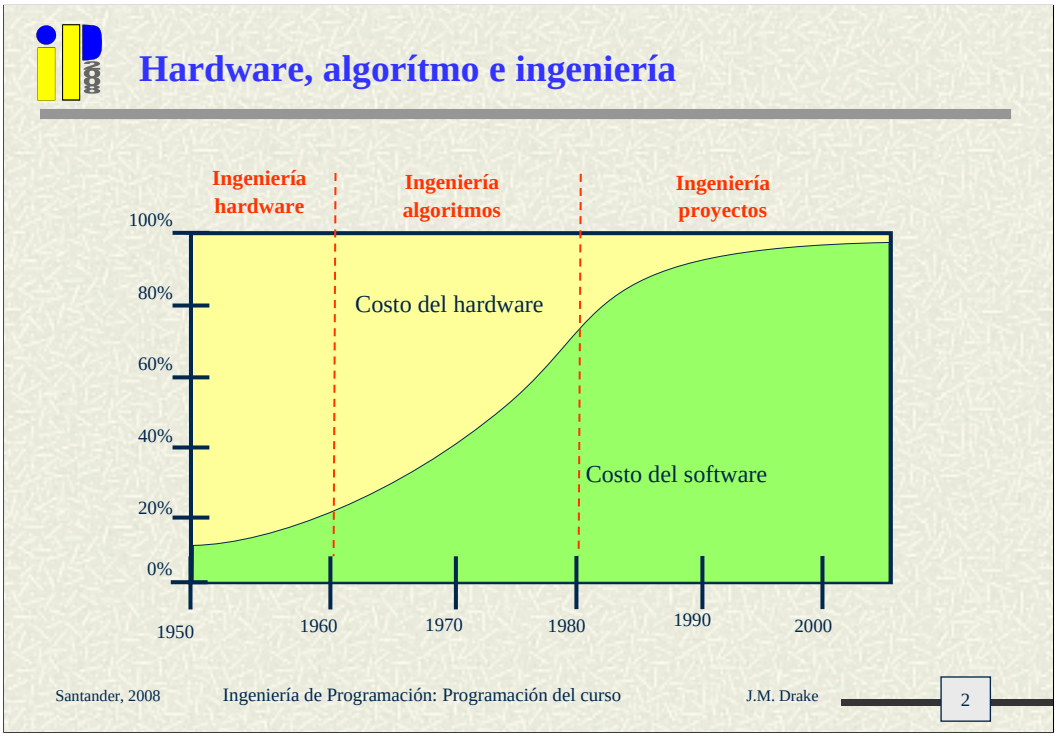
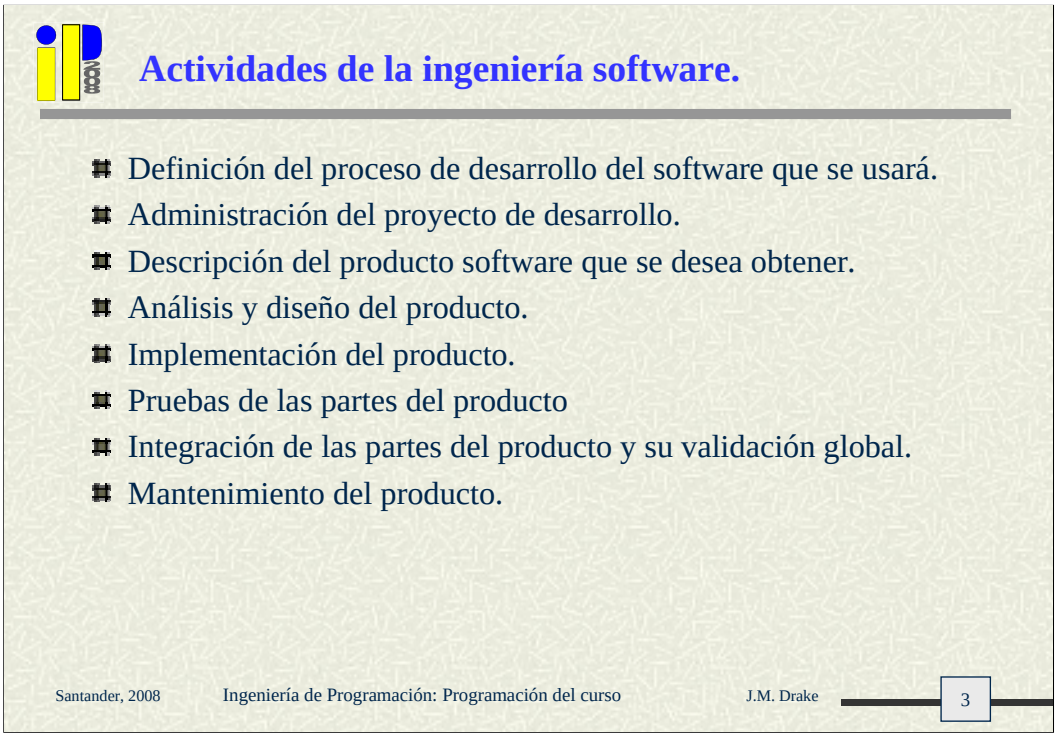

Ingeniería Software
4º Físicas
Programación del curso
José M. Drake (drakej@unican.es)
Patricia López Martínez (lopezpa@unican.es)
Computadores y Tiempo Real
Santander, 2008

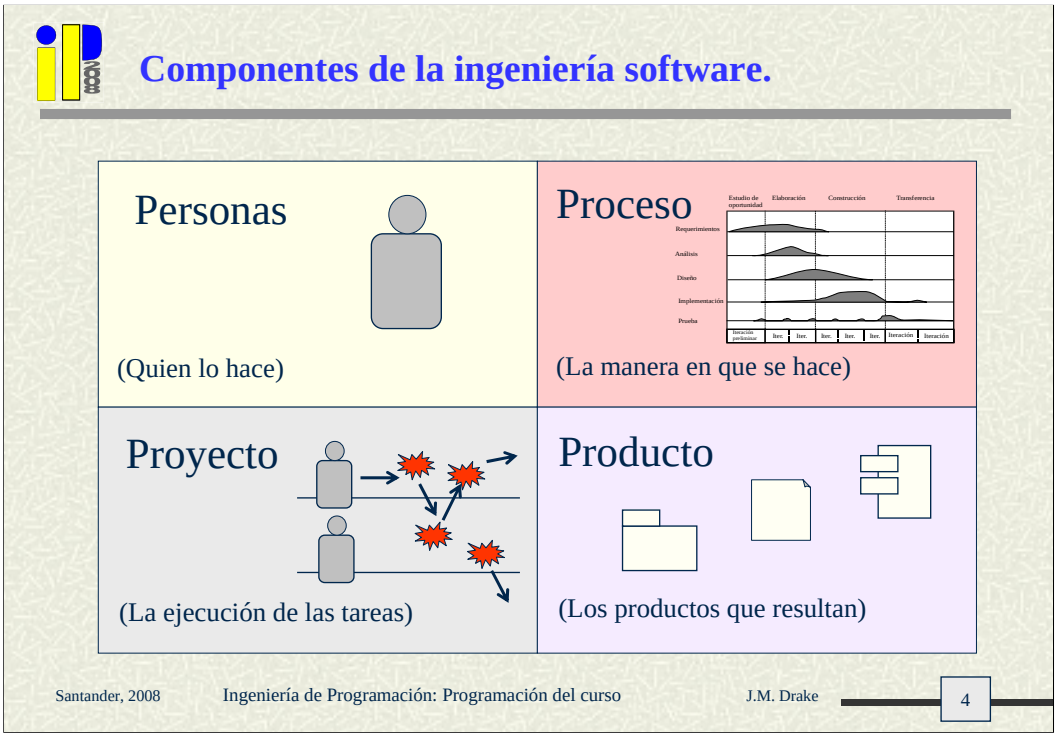


La ingeniería software es por definición un tipo de ingeniería y, por tanto, tiene el mismo conjunto de responsabilidades sociales que las otras.
¿En que se diferencia la ingeniería de software de los otros tipos de ingenierías y en que es similar?
•Laingeniería de software comparte con las otras ingenierías la necesidad de describir exhaustivamente y metódicamente lo que tiene que producirse (análisis de requisitos).
•Los proyectos software están sujetos a cambios frecuentes que incluyen los impuestos durante el desarrollo del producto.


Hardware, algorítmo e ingeniería
Ingeniería hardware
Ingeniería algoritmos
Ingeniería proyectos
100%
80%
60%
40%
20%
0%
Costo del hardware
Costo del software
1950
1960
1970
1980
1990
2000
Santander, 2008
Ingeniería de Programación: Programación del curso
J.M. Drake
2



En la década de los 90, dos tendencias se impusieron en el desarrollo del software: Una fue el crecimiento explosivo de las aplicaciones y de sus dimensiones, la otra, el nacimientos de nuevas herramientas y paradigmas conceptuales basadas en la programación orientada a objetos.
Sin embargo, a pesar de las nuevas tendencias, las actividades básicas han permanecido estables. Estas actividades son las que se describen en la transparencia de arriba.



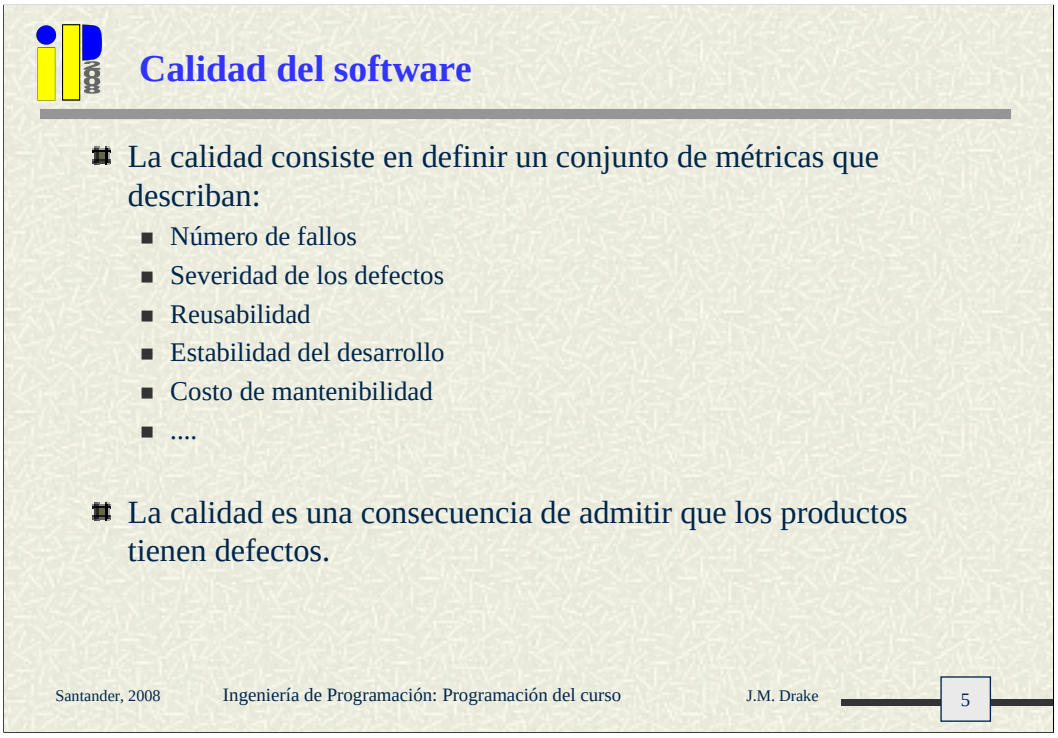
La ingeniería software incluye Personas, Proceso, Proyecto y Productos.

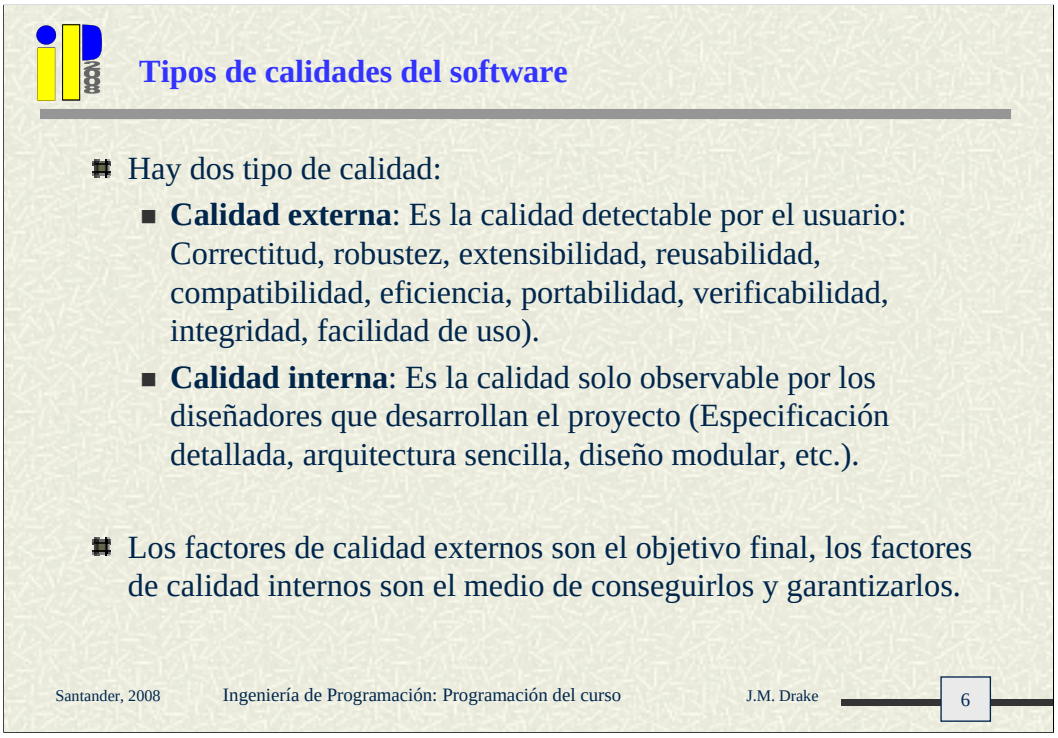
Personas: Una aplicación software de mediana complejidad requiere la participación de un conjunto de personas, algunos son técnicos que desarrollan los productos, otros son los interesados (personas que ganan o pierden algo con el éxito de la aplicación). Es importante definir las funciones y responsabilidades de los participantes para que el trabajo alcance su éxito.

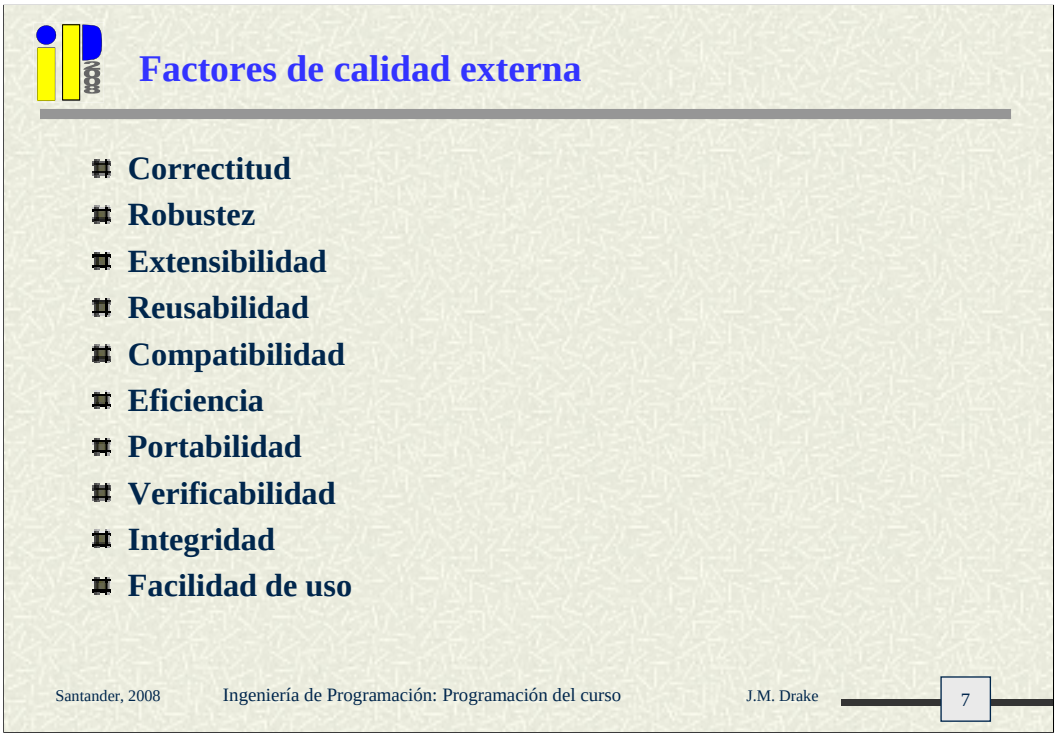
Proceso: Representa la **organización** de las actividades que hay que realizar y los productos que deben obtenerse para elaborar la aplicación. Se han propuesto muchos tipos de proceso, algunos están definidos de acuerdo con las herramientas de que se disponen, otros están definidos en función del tipo de aplicación que se desarrolla. En este curso nos centraremos en el proceso USPD (Unified Software Development Process).

Proyecto: Es el conjunto de de **actividades** necesarias para producir los artefactos requeridos. Incluye: Negociar con el cliente, escribir la documentación, formular el diseño, escribir el código, y probar el producto. La programación orientada a objetos (**OO**) es un paradigma muy útil para el desarrollo del proyecto, en particular para facilitar los cambios entre fases ya que se hace usando las mismas abstracciones del dominio. El lenguaje unificado de Modelado (**UML**) es una herramienta muy útil ya que permite representar los conceptos de forma gráfica y facilita el intercambio de ideas entre los participantes.

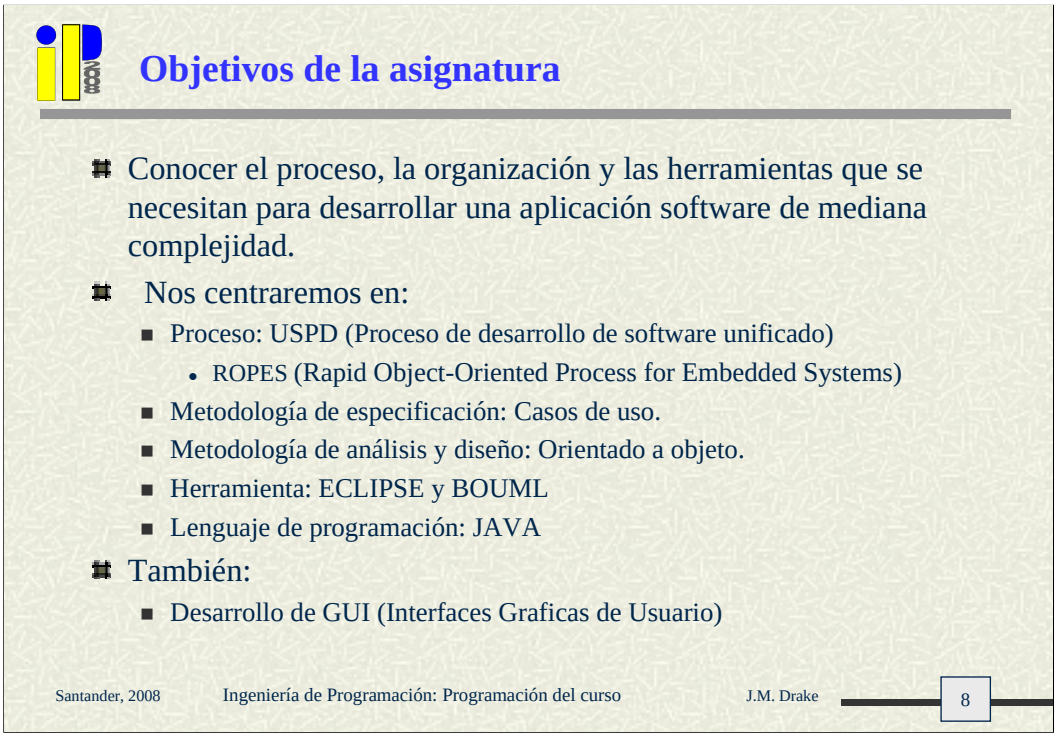
Producto: El producto de una aplicación software no es sólo el código ejecutable, sino todos lo elementos (artefactos) que lo describen y que permiten su compresión, manejo, uso mantenimiento, etc. Entre los principales productos están: Los documentos de especificación de requisitos, que describen la arquitectura hardware y software, que describen el diseño detallado de cada módulo, las implementaciones o código entregable y los procedimientos de prueba y validación.


Calidad del software
La calidad consiste en definir un conjunto de métricas que describan:
Número de fallos
Severidad de los defectos
Reusabilidad
Estabilidad del desarrollo
Costo de mantenibilidad
....
La calidad es una consecuencia de admitir que los productos tienen defectos.
Santander, 2008
Ingeniería de Programación: Programación del curso
J.M. Drake
5

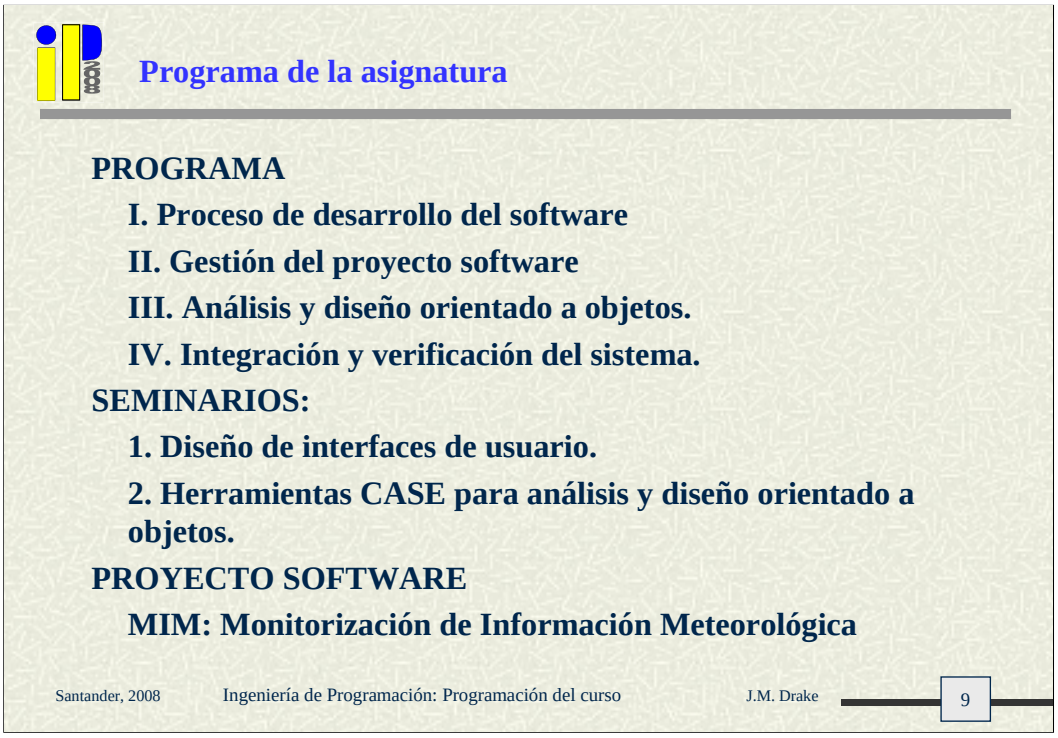

Tipos de calidades del software
Hay dos tipo de calidad:
Calidad externa: Es la calidad detectable por el usuario: Correctitud, robustez, extensibilidad, reusabilidad, compatibilidad, eficiencia, portabilidad, verificabilidad, integridad, facilidad de uso).
Calidad interna: Es la calidad solo observable por los diseñadores que desarrollan el proyecto (Especificación detallada, arquitectura sencilla, diseño modular, etc.).
Los factores de calidad externos son el objetivo final, los factores de calidad internos son el medio de conseguirlos y garantizarlos.
Santander, 2008
Ingeniería de Programación: Programación del curso
J.M. Drake
6

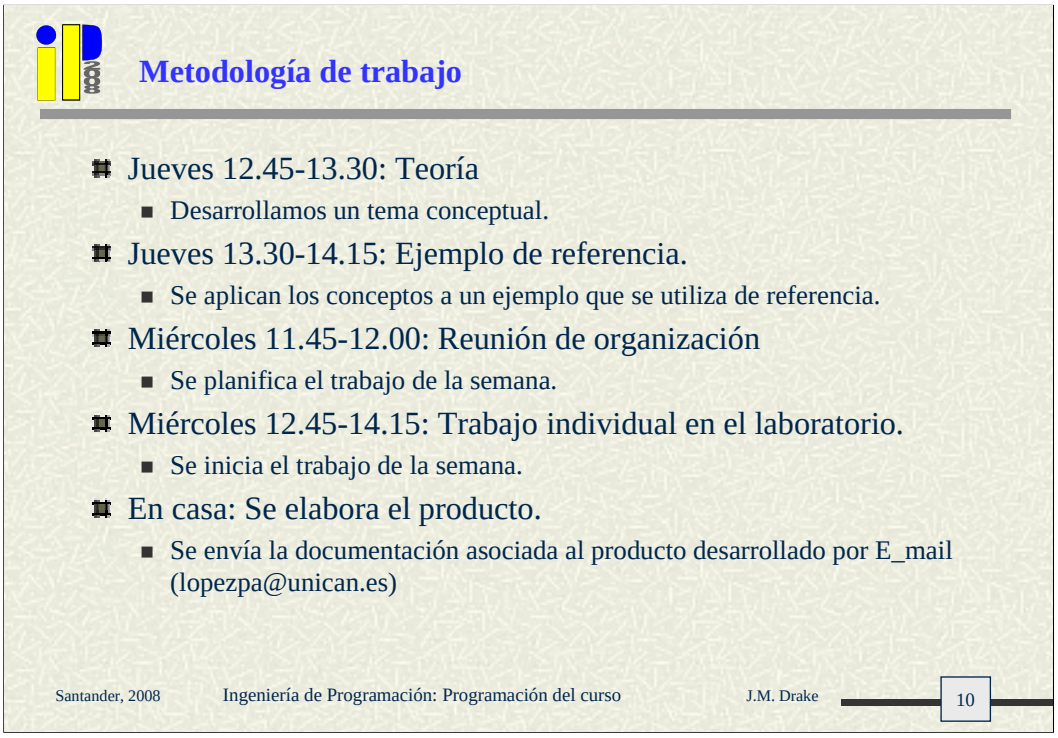


CORRECTITUD (Correctness): Es el factor de calidad externo fundamental. Es un factor muy fácil de definir, pero solo se puede verificar si las especificaciones son exhaustivas.
ROBUSTEZ (Robustness): La respuesta fuera de la especificación no es de interés, pero la aplicación debe dar respuestas correctas cuando se retorna a condiciones especificadas. El concepto de robuztez es difuso, se suele requerir que nunca se alcancen estados catastróficos.
EXTENSIBILIDAD (Extendebility): Aparentemente es consustancial con el concepto de software, pero en los proyectos grandes no es posible estimar los efectos colaterales de los cambios. Las dos bases de la extensibilidad son: La simplicidad del diseño y el diseño descentralizado.
REUSABILIDAD (Reusability): Es un concepto importante en la industria del software. Debería se facil de alcanzar ya que son pocas las operaciones básicas de software, sin embargo aún no es así.
COMPATIBILIDAD (Compatibility): Las claves para conseguir la compatibilidad son: la homogeneidad y la estandarización.
EFICIENCIA (Efficency): Es la capacidad de un producto software de hacer un uso completo de los recursos que le proporciona la plataforma sobre la que se ejecuta.
PORTABILIDAD (Portability) : Es la capacidad de un producto software para ser transferido a diferentes plataformas hardware y entornos software.
VERIFICABILIDAD (Verifiability): Es la facilidad que proporciona un software para detectar y trazar fallos durante las fases de validación y operativa.
INTEGRIDAD (Integrity): Es la capacidad que ofrece un producto software para proteger su recursos (código, datos, ficheros, etc.) contra accesos y modificaciones no autorizadas.
FACILIDAD DE USO (Ease of use): Es la facilidad que presenta el producto software para ser utilizada, ser gestionada, preparar la información de entrada, interpretar los resultados, etc.



El objetivo de la asignatura es aprender a desarrollar una aplicación software de mediana complejidad.
Entendemos por aplicación de mediana complejidad aquella que puede ser desarrollada por un equipo de 3 o 4 personas durante un semestre, lo que equivale a un código fuente de unas (4 personas*120 días*25 líneas = 12.000 Líneas de código fuente JAVA sin comentarios (NCSLOC)) (72.000 €).
El desarrollo de un proyecto implica diferentes aspectos:
-Requiere utilizar un **proceso de desarrollo** dentro del cual se encuadren las diferentes tareas que conducen a la elaboración de una aplicación.
-Requiere conocer la organización de un **equipo de personas**, identificar los papeles que han de desempeñar y sus responsabilidades.
-Requiere conocer los **productos** que deben generarse de los que uno es el código ejecutable, pero que también requiere documentación textual y gráfica relativa a la especificación, análisis, diseño, codificación, verificación, validación y manuales de usuario.
-Requiere conocer las **herramientas** que permiten describir y modelar las ideas y los conceptos de forma humana no basada en lenguajes de programación.
El curso no está planteado como un planteamiento genérico de metodologías de desarrollo de software, sino centrado en un conjunto concreto que se va a utilizar.


Programa de la asignatura
PROGRAMA
I. Proceso de desarrollo del software
II. Gestión del proyecto software
III. Análisis y diseño orientado a objetos.
IV. Integración y verificación del sistema.
SEMINARIOS:
1. Diseño de interfaces de usuario.
2. Herramientas CASE para análisis y diseño orientado a objetos.
PROYECTO SOFTWARE
MIM: Monitorización de Información Meteorológica
Santander, 2008
Ingeniería de Programación: Programación del curso
J.M. Drake
9

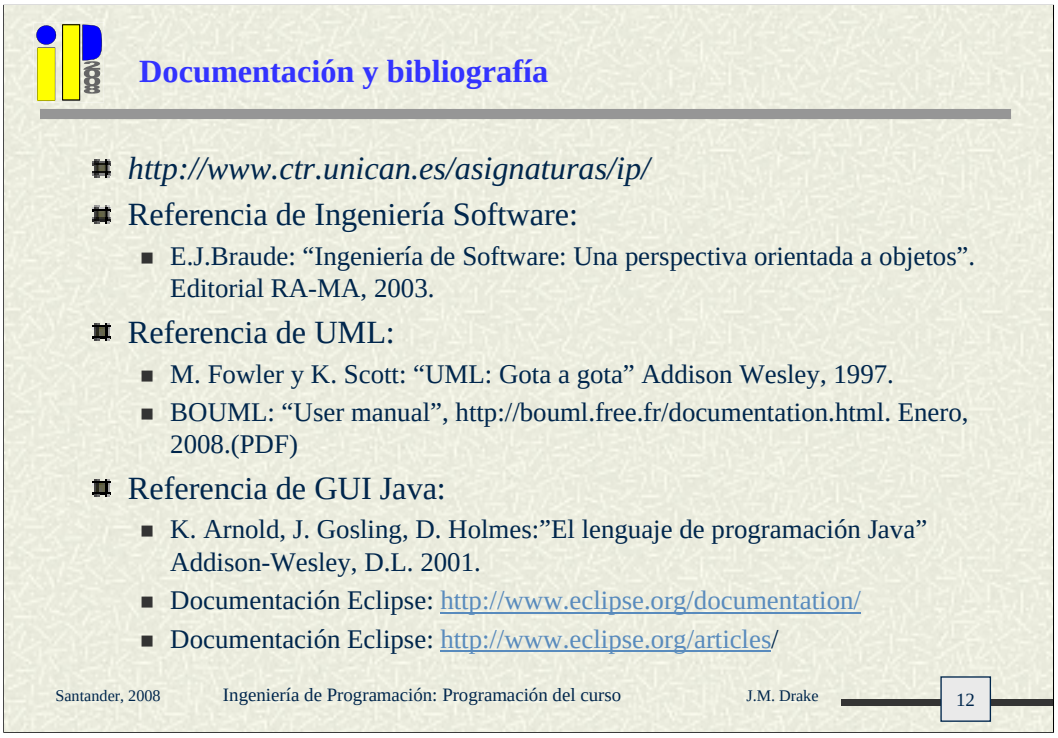


La **organización del trabajo** va a tener una planificación con ciclo semanal:
45minutos: En una sesión de clase magistral se propone los conceptos relativos a algún tema de forma sistemática y ordenada.
45 minutos: En una sesión de clase conjunta se aplican los conceptos a un ejemplo que se irá desarrollando a lo largo del curso y que se tomará como referencia.
30 minutos: En una reunión del equipo se organiza el plan de trabajo de la semana.
120 minutos: En el laboratorio se inicia el trabajo planificado, y trabajando sobre él se concreta y termina de definir.
En casa: Se termina el trabajo semanal que cada uno tiene asignado, se elabora la documentación relativo a la tarea y se remite al profesor por E_mail (drakej@unican.es) con una antelación de 24 horas a la reunión de planificación de trabajo de la siguiente semana.

Papeles en el proyecto

- El profesor:
  - Cliente
  - Administrador del proyecto
  - Lider
- El alumno:
  - Desarrollador
  - Control de calidad.

Proceso de desarrollo del software: Componentes del desarrollo. Modelo de proceso en cascada. Modelo de proceso en espiral. Modelo USDP. Documentación. Calidad del software.
Administración del proyecto: Componentes de la administración de un proyecto. Organización del personal. Identificación y eliminación de los riesgos. Herramientas. Planificación y calendario.
Herramientas: Instalación de Rational Rose y J_Builder.
Especificación del proyecto y análisis de requisitos: Clasificación de los requisitos. Requisitos orientados al cliente. Descripción y documentación de los requisitos de cliente. Prototipos rápidos y GUIs. Requisitos orientados a los desarrolladores. Requisitos funcionales y no funcionales. Diagramas de contexto. Diagramas de secuencias y de colaboración. Documentación de requisitos.
Análisis y diseño de clases y arquitectura: Análisis orientado a objetos. Modelo estructural. Subsistemas, componentes, clases e interfaces. Arquitectura software y hardware. Marcos de referencia y patrones de arquitectura.



La documentación del curso se entregará a los alumnos directamente en clase, o se colgará de la página www.ctr.unican.es/IngenieriaSoftware/.

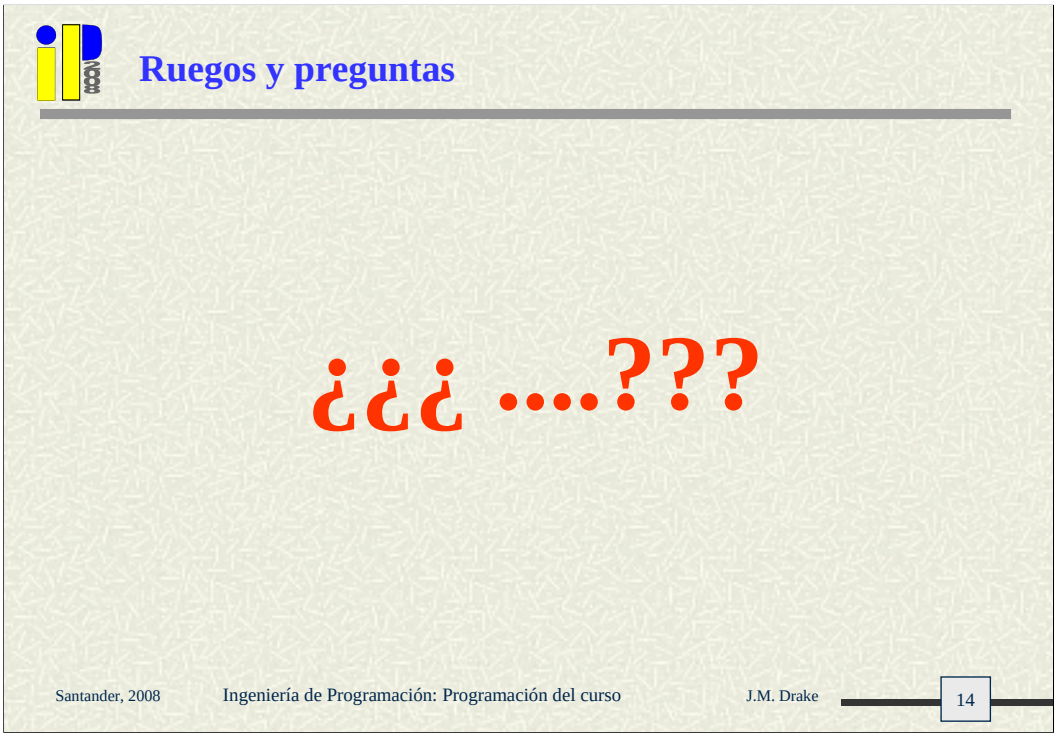
El alumno no requerirá ningún libro de texto para seguir el curso. Los libros de texto siguientes se utilizarán para organizar el curso.
-E.J. Braude: Ingeniería de software: Una perspectiva orientada a objetos." Es el libro de texto que va a seguirse en la asignatura. Es un muy buen libro de texto, pero sobrepasa los objetivos del curso. Caso de que quiera tener un libro de ingeniería software, en este momento, éste es el mejor.
- M. Fowler y K. Scott: "UML: Gota a gota".
Es un libro sobre UML para principiante. Es muy claro y sencillo. No lo compre ya que en es muy básico y en seguida resulta inútil.
- Rational: "Rational Rose 2000e using RoseJ" .
Es un documento sobre la correspondencia en los modelos UML y código fuente Java que lleva a cabo el generador de código de Rational Rose. Se utilizará como manual. (PDF que se colgará de la página WEB del curso)
- Borland: "Diseño de aplicaciones con Jbuilder" (PDF)
Se utilizará como manual cuando se aborde el diseño de las GUIs.
- Y.D. Liang: "Rapid Java Application development using J Builder"
Libro de texto relativo al uso de Jbuilder como herramienta de desarrollo de aplicaciones Java. Tiene buenos ejemplos.

Seguimiento y evaluación de la asignatura.

- El curso tiene por objetivo seguir un ciclo completo de desarrollo de una aplicación y por tanto requiere una atención y dedicación continua semana a semana.

  La perdida de clases por cualquier razón debe ser recuperada.

- La evaluación se realiza por los documentos que semanalmente se entregan sobre el trabajo de equipo.
  - Existe una evaluación final para establecer la calificación.
  - Caso de que no se esté conforme con la calificación que resulte de la evaluación continua, puede realizar una evaluación escrita.


Ruegos y preguntas
¿¿¿ ....???
Santander, 2008
Ingeniería de Programación: Programación del curso
J.M. Drake
14