Evolución de la Programación Orientada a Objetos

Dr. Luis Gerardo de la Fraga

Departamento de Computación
Cinvestav
Correo-e: fraga@cs.cinvestav.mx

7 de diciembre de 2006

Motivación

Se presentará en esta charla:

- Una revisión histórica del paradigma OO
- Situar OO contra otros paradigmas, en especial los lenguajes de muy alto nivel
- Una crítica personal a la programación OO

Contenido

- Introducción
- Antecedentes de la programación OO
- Características del modelo OO
- Comparación entre lenguajes OO
- Clasificación de los metodologías OO
- Los lenguajes de muy alto nivel
- ¿Cuál es el mejor paradigma?

Introducción (1/3)

- La programación OO empezó hace 30 años
- En los 1990s se incrementó dramáticamente la demanda para sistemas de software OO, por la promesa en la revolución en el desarrollo de software.
- Han aparerecido varias metodologías para el desarrollo de software, que tienen que ver con algunas o todas las fases del ciclo de vida del software, desde los requerimientos al mantenimiento.

Introducción (2/3)

Algunas características importantes de los sistemas de software actuales son:

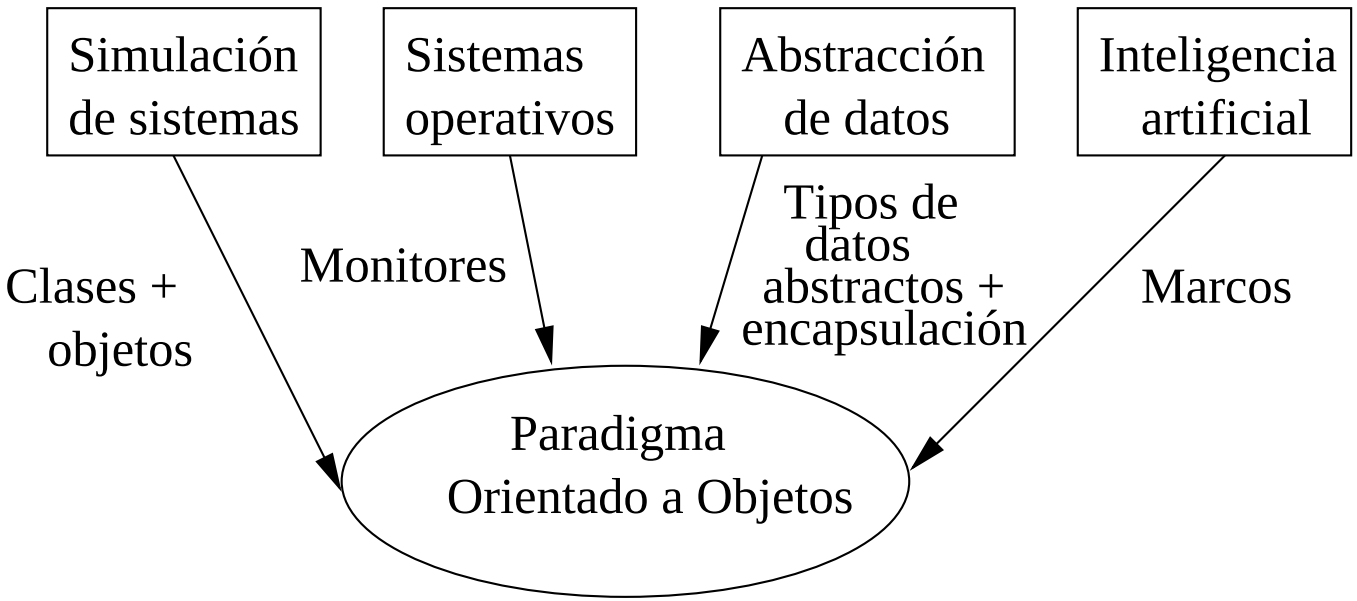
- *Complejidad*: la arquitectura interna de los sistemas actuales de software es compleja, incluyen frecuentemente concurrencia y paralelismo. La abstracción en terminos de conceptos de OO es una técnica que ayuda a tratar con la complejidad.
- *Amigabilidad*: Este es un requerimiento de suma importancia para los sistemas de software en general.
- *Reusabilidad*: Tomar componentes creados por otros es mejor que crearlos nuevos. La herencia es un mecanismo de OO que estimula la reusabilidad del software. Facilita el rápido desarrollo del software.

Introducción (3/3)

Las razones del rápido desarrollo en los últimos 15 años han sido:

- Una mejor modelación de aplicaciones del mundo real
- La posibilidad del reuso del software durante el desarrollo de un sistema de software

Antecedentes (1/2)



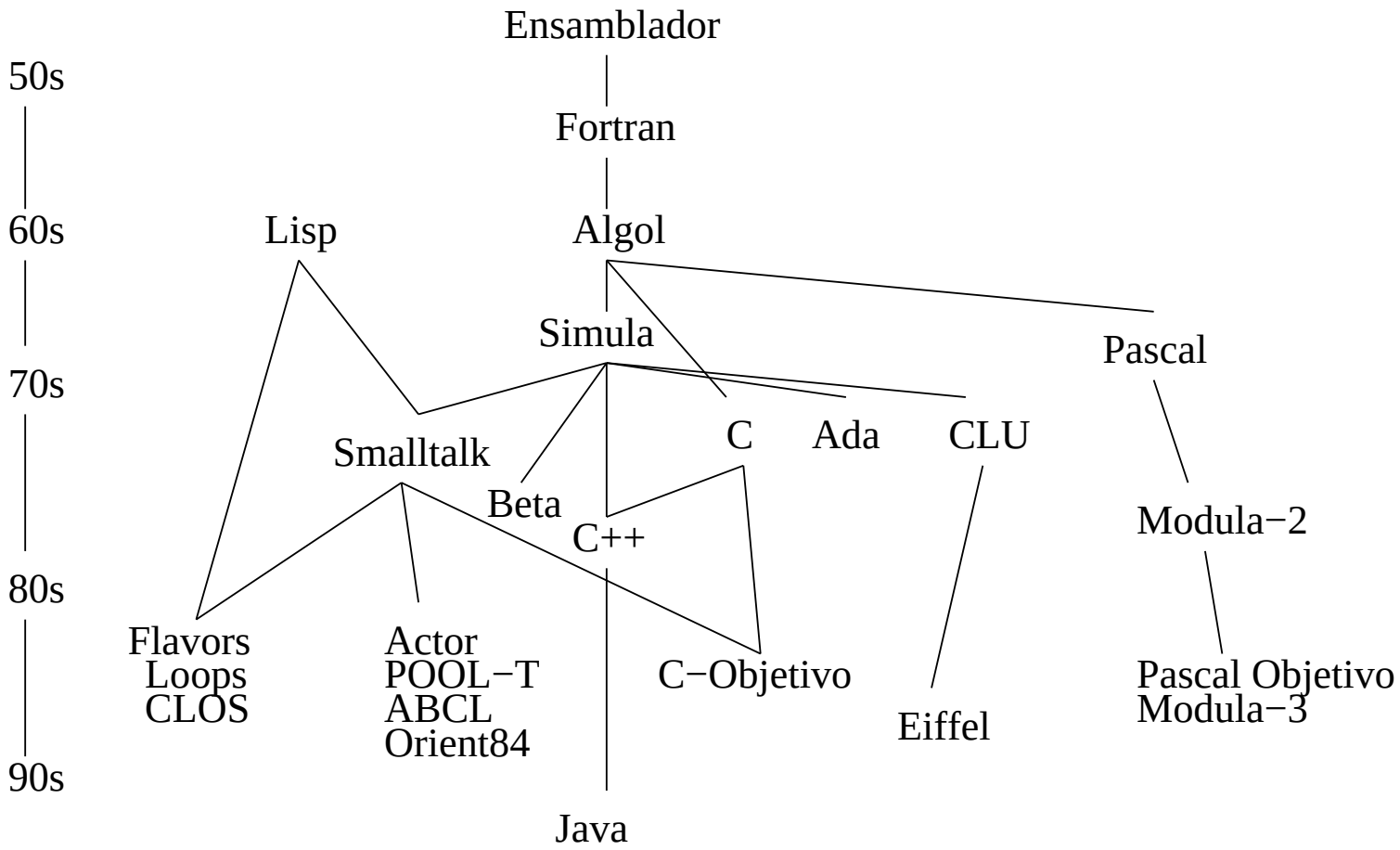
Antecedentes (2/2)

La característica común de estas ideas es que un objeto es una entidad lógica o física que está autocontenida.

Características

- OO se define por herencia, encapsulación, métodos y mensajes, como en Smalltalk.
- OO se define encapsulación, abstracción de datos, métodos, mensajes, herencia y vinculación dinámica.
- Es un modelo de simula el comportamiento ya sea de una parte del mundo real o imaginario.
- Objetos, clases y herencia. Los objetos son entidades autónomas que tienen un estado y responden a mensajes. Las clases agrupan los objetos por sus atributos y operaciones.
- Todas tienen el común de usar objetos como una encapsulación para proteger los datos con todas las operaciones legales que actuan sobre esa información oculta.

Comparación entre lenguajes OO (1/4)



Comparación entre lenguajes OO (2/4)

Lenguaje ensamblador:

- Intrucciones de máquina (operadores) que manipulan en contenido de localidades de memoria (operandos)

Lenguaje de alto nivel:

- Operadores se vuelven declaraciones y los operandos en variables y estructuras de datos.

Los programas están compuestos de de una colección de variables que representan algún dato y un conjunto de procedimientos que manipulan esas variables.

Comparación entre lenguajes OO (3/4)

Características X Lenguajes	Tipos de datos	Apoyo de herencia	Vinculación dinámica	Biblioteca extensa
Simula	Si	Si	Si	No
CLU	Si	No	Si	No
Ada	Si	No	No	Si
Smalltalk	Si	Si	Si	Si
C Objetivo	Si	Si	Si	Si
C++	Si	Si	Si	Si
CLOS	Si	Si	Si	No
Pascal Obj.	Si	Si	Si	No
Beta	Si	Si	Si	No
Eiffel	Si	Si	Si	Si
Actor	Si	Si	Si	No
Java	Si	Si	Si	Si

Clasificación de los metodologías OO (1/6)

Varios métodos han sido propuestos para sistematizar el proceso de vida del software. Y muchas metodologías de desarrollo de software han sido propuestas, y éstas pueden clasificarse en tres categorías:

- Descomposición funcional.
- Enfásis en datos, más que en funciones.
- Ambos puntos de vista: funcional y datos.

Clasificación de los metodologías OO (2/6)

- Descomposición funcional:
  Diseño Estructural, Refinamiento por Pasos.
- Enfásis en datos, más que en funciones:
  Programación Estructurada, Modelo Entidad-Relación
- Ambos puntos de vista: funcional y datos:
  Análisis Estructural, Análisis de Sistemas Estructurados,
  Análisis de Sistemas Estructurados y Metodología de Diseño.

Clasificación de los metodologías OO (3/6)

Desarrollo estructural:
Análisis estructural, diseño estructural y programación estructurada.

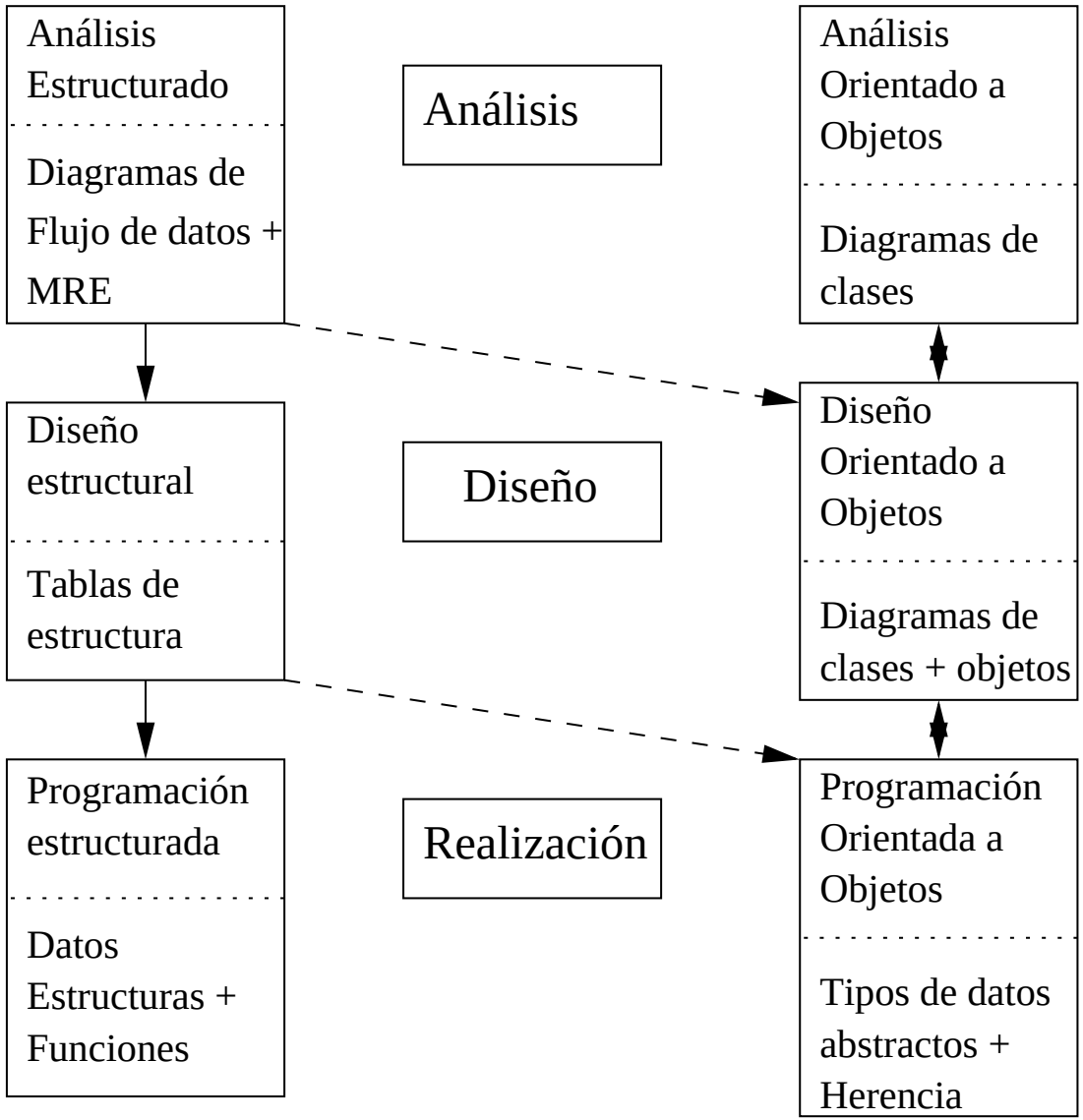
Aplicar primero el diseño estructural y luego la proximación orientada a objetos resulta en problemas dado que no se puede mapear apropiadamente las funciones en objetos.

Clasificación de los metodologías OO (4/6)

Metodologías "orientadas a objetos":

- **Adaptación**: mezclar una aproximación orientada a objetos con una metodología bien conocida de desarrollo estructural.
- **Asimilación**: usar una metodología orientada a objetos para desarrollar sistemas de software, pero que siguen el modelo tradicional del ciclo de vida del software.

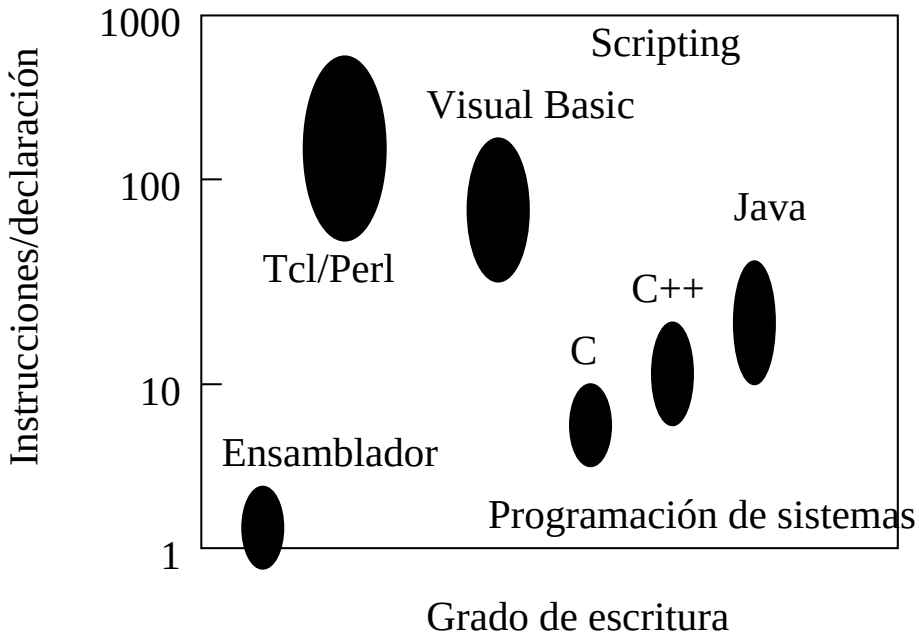
Clasificación de los metodologías OO (5/6)



Clasificación de los metodologías OO (6/6)

La orientación a objetos tiene la necesidad de una vista organizada y manejable del desarrollo del software en todas las fases del modelo del ciclo de vida del software.
Esta demanda ha sido satisfecha por el Lenguaje de Modelado Unificado (UML) y por herramientas CASE tales como Rational Rose.

Son lenguajes que trabajan en scripts, son de más alto nivel con menos escritura



Ejemplos: Perl, PHP, Python, Tcl/Tk, Visual Basic

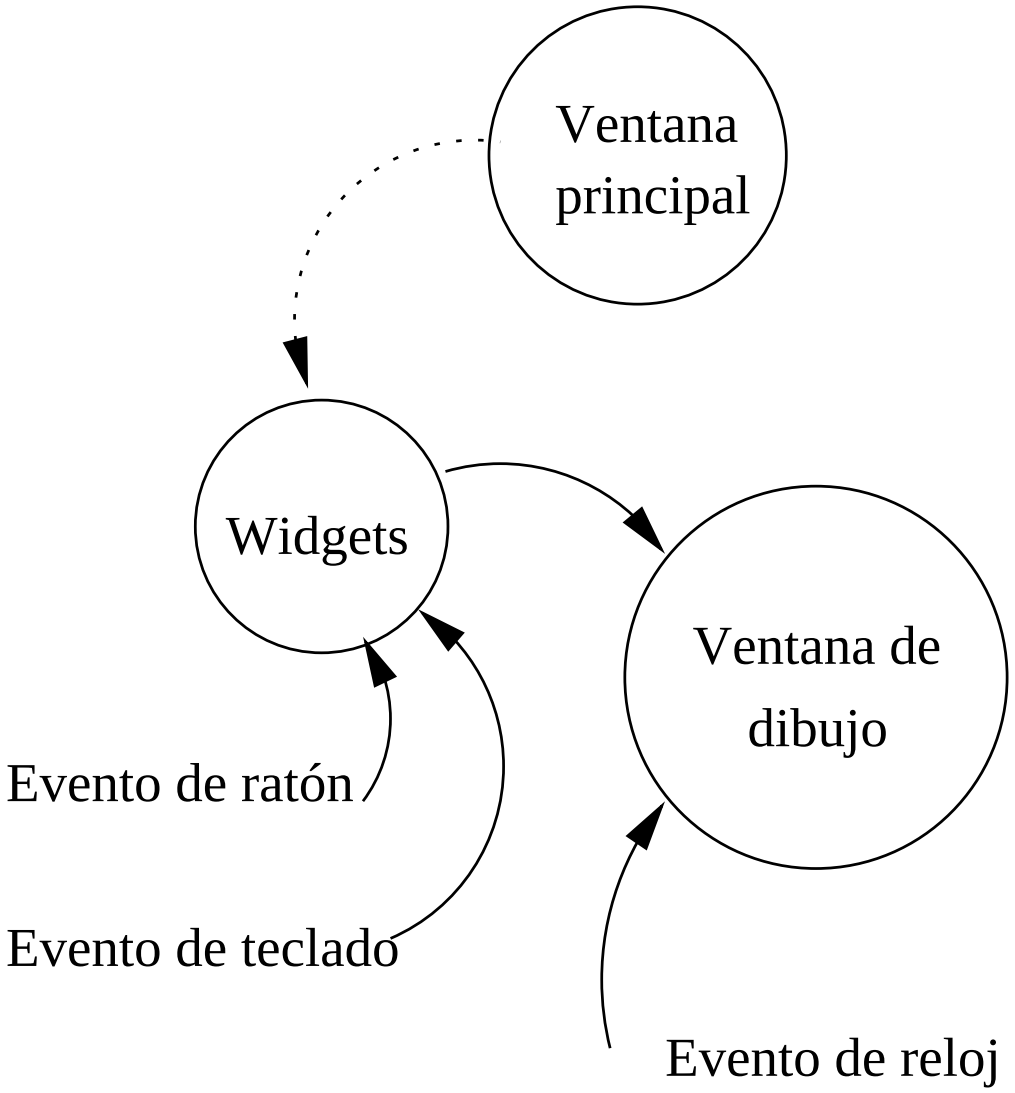
- Puede verse como la unión de: el shell, awk y sed.
- También permite el uso de objetos.
- Tiene una gran cantidad de “módulos”: www.cpan.org

¿Cuál es el mejor paradigma?

- Procedural
- Orientado a objetos
- Alto nivel
- Deben conocerse y dominarse todos los paradigmas

Realizaciones

- Los lenguajes de alto nivel pueden usar en programas cortos, de rápido desarrollo, que se ejecutarán unas pocas veces. También en programas para tratamientos de textos.
- Los procedurales en programas pequeños donde queremos muy alta eficiencia.
- OO debe usarse en grandes proyectos de software donde se tiene una clara identificación de los objetos. Para realizar interfaces de usuario.


Ventana
principal
Widgets
Ventana de
dibujo
Evento de ratón
Evento de teclado
Evento de reloj

Realizaciones

- ¿Qué tanto es grande?
- XMIPP se cambió a objetos en tres años.
- El sistema de administración de conferencias se hizo procedural.
- ¿Se puede forzar el uso de objetos?

Aplicación	Comparación	Razón de código	Razón de esfuerzo
Aplicación de base de datos	Ver. C++: 2 meses: Ver. TCL: 1 día		60
Instalación y prueba de un sistema de cómputo	Aplicación de prueba en C: 272,000 lineas, 120 meses, Aplicación C FIS: 90,000 lineas, 60 meses. Versión Tcl/Perl: 7,700 lineas, 8 meses	47	22
Biblioteca de base de datos	Versión en C++: 2-3 meses Versión Tcl: 1 semana		8-12
Escaner de seguridad:	Versión en C: 3,000 lineas, Ver. Tcl: 300 lineas	10	
Simulador e interfaz	Versión en Java: 3,400 lineas, 3-4 semanas, Versión en Tcl: 1,600 lineas, $<$ 1 semana	2	3-4

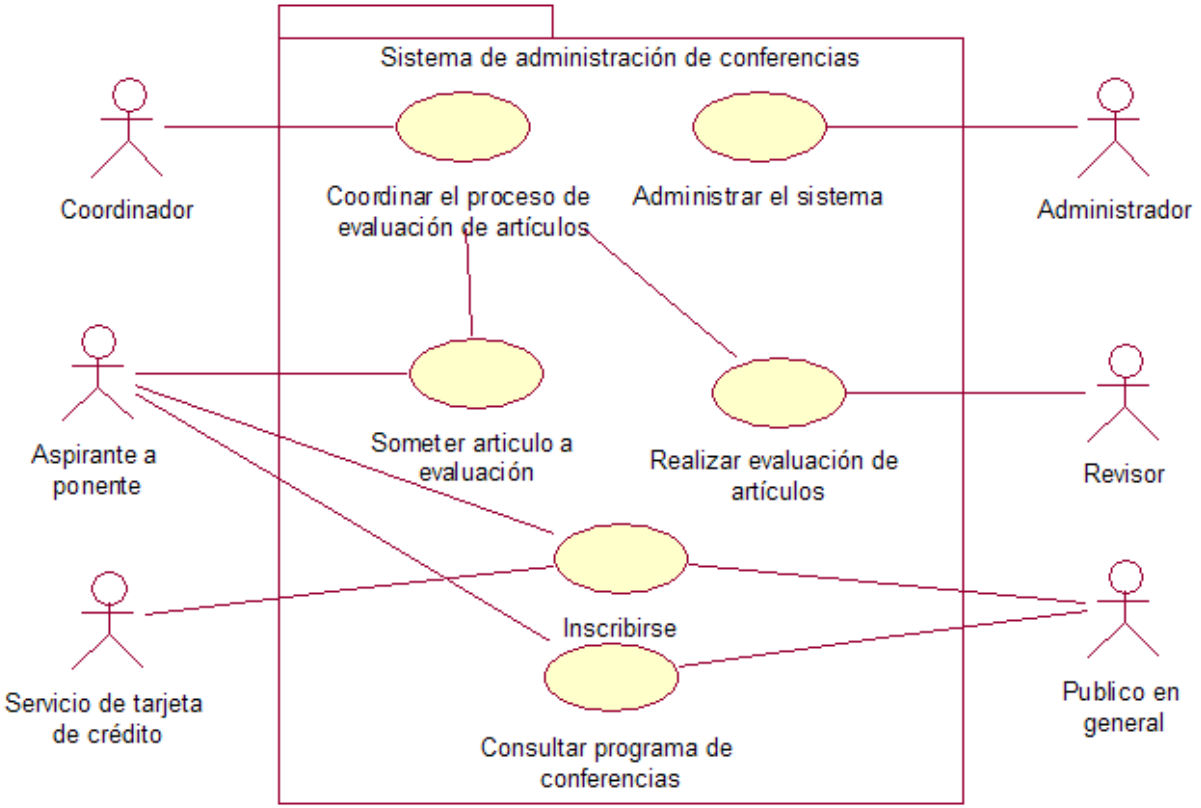
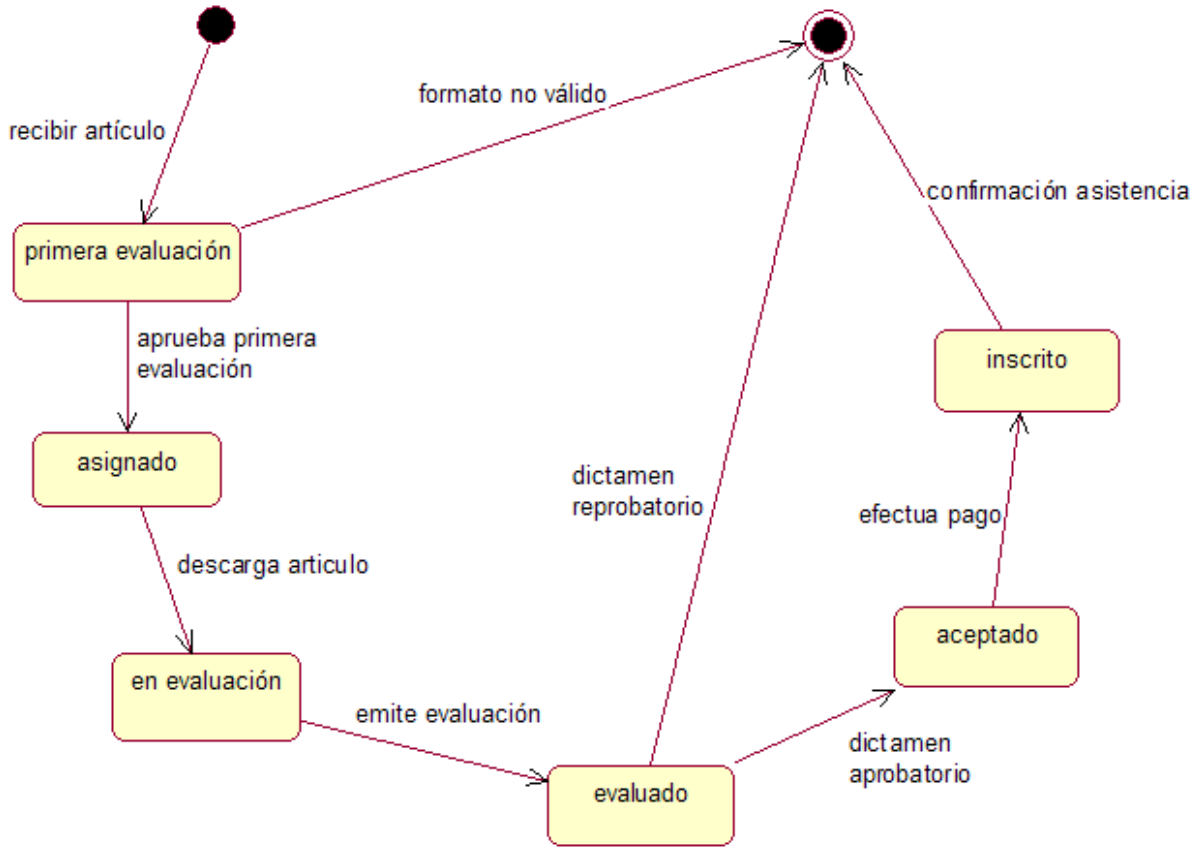


Diagrama general de casos de uso

Diagrama de estados de un artículo



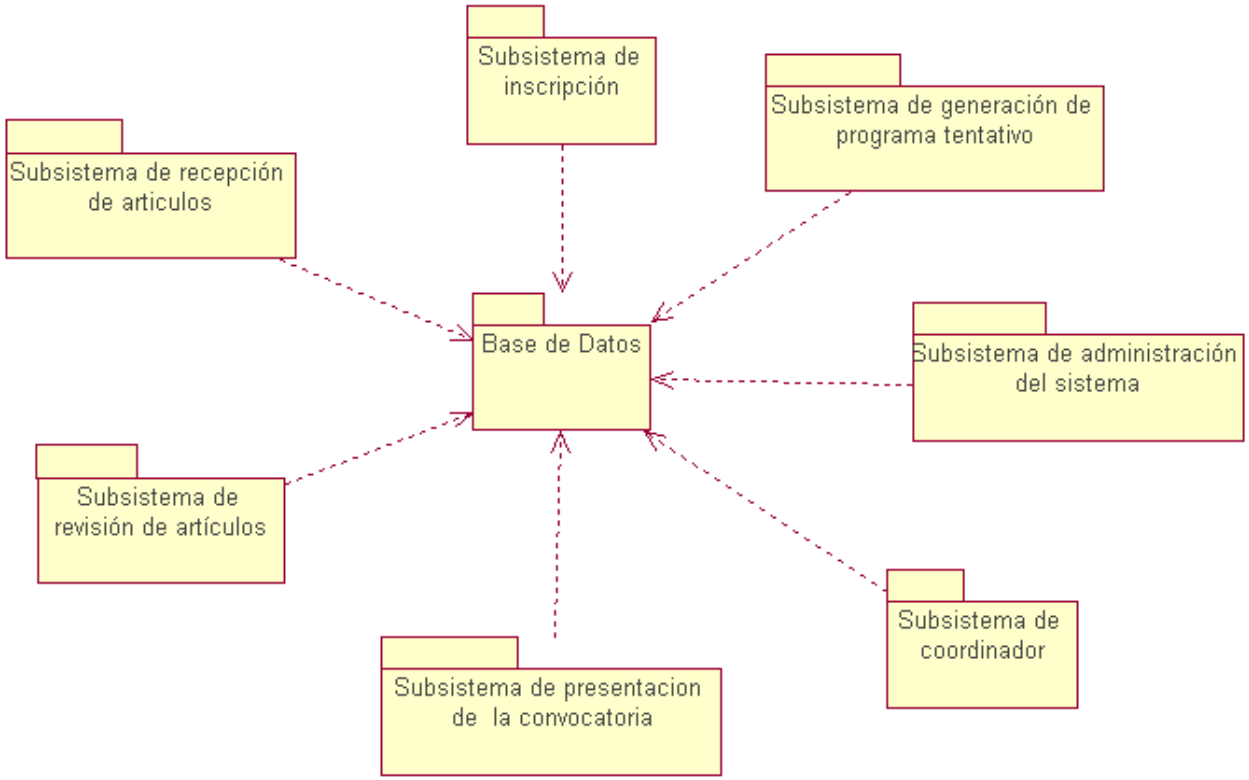
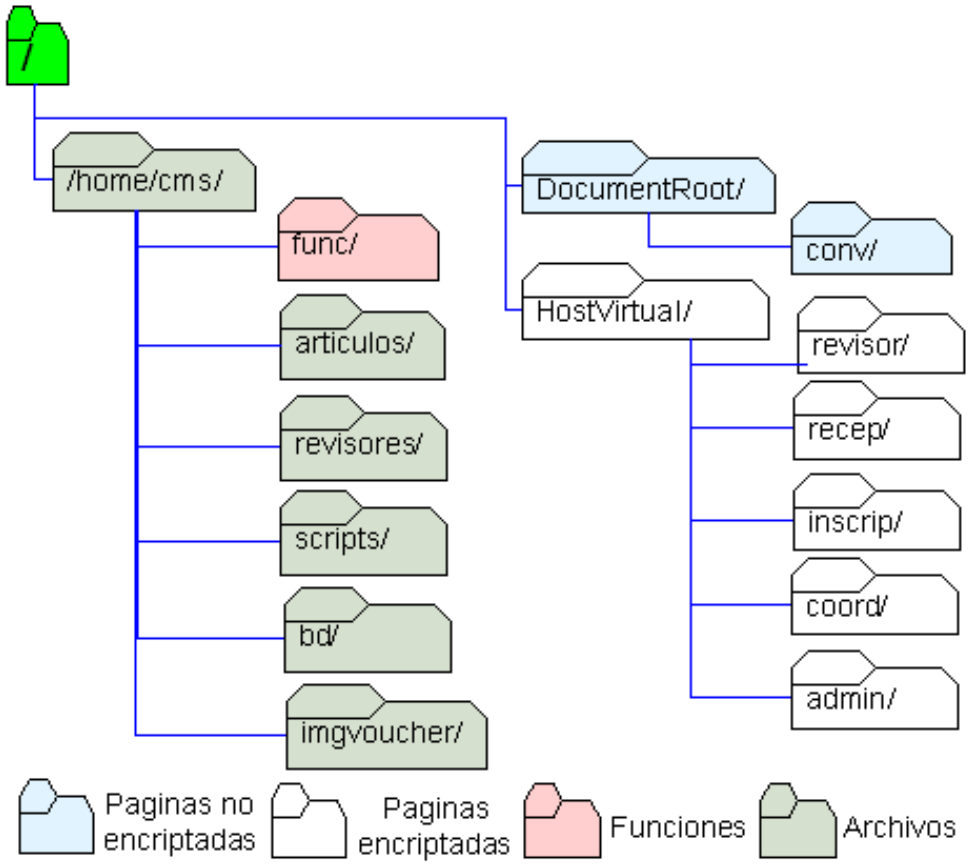


Diagrama de subsistemas

Implementación: Disposición física de los archivos



Disposición física de los archivos

Conclusiones

- La programación OO tuvo un crecimiento muy grande en los 1990s
- Aunque todavía se debate los beneficios de OO, a treinta años de su invención, OO todavía seguirá usándose
- Para usar OO deben aplicarse las técnicas de Ingeniería de Software.
- Deben dominarse los tres paradigmas: alto nivel, OO, procedural.

Esta presentación puede encontrarse en:

http://delta.cs.cinvestav.mx/~fraga/Charlas/

Referencias:

- L.F. Capretz, A brief history of the Object-Oriented Approach, Software Engineering Notes (ACM SIGSOFT), pp 1-10, vol 28, no 2, March 2003
- J.K. Ousterhout, Scripting: Higher level programming for the 21st century, IEEE Computer Magazine, pp 23-30, March 1998.

Ofrecemos:

1. Maestría en Ciencias en Computación
2. Doctorado en Ciencias en Computación

http://www.cs.cinvestav.mx