Introducción a la Tecnología Orientada a Objetos

UNEG

La Crisis del Software

- Fenómeno ocurrido en la década de los 80's como consecuencia del auge de la automatización haciendo uso de computadoras.
- La mayoría del software se construye a la medida, en vez de ensamblar componentes existentes.
- Los programadores no disponen / usan "componentes" de software reutilizables → empezar desde cero.
- "Complejidad del Software"

  "… es una propiedad esencial, no accidental" Brooks

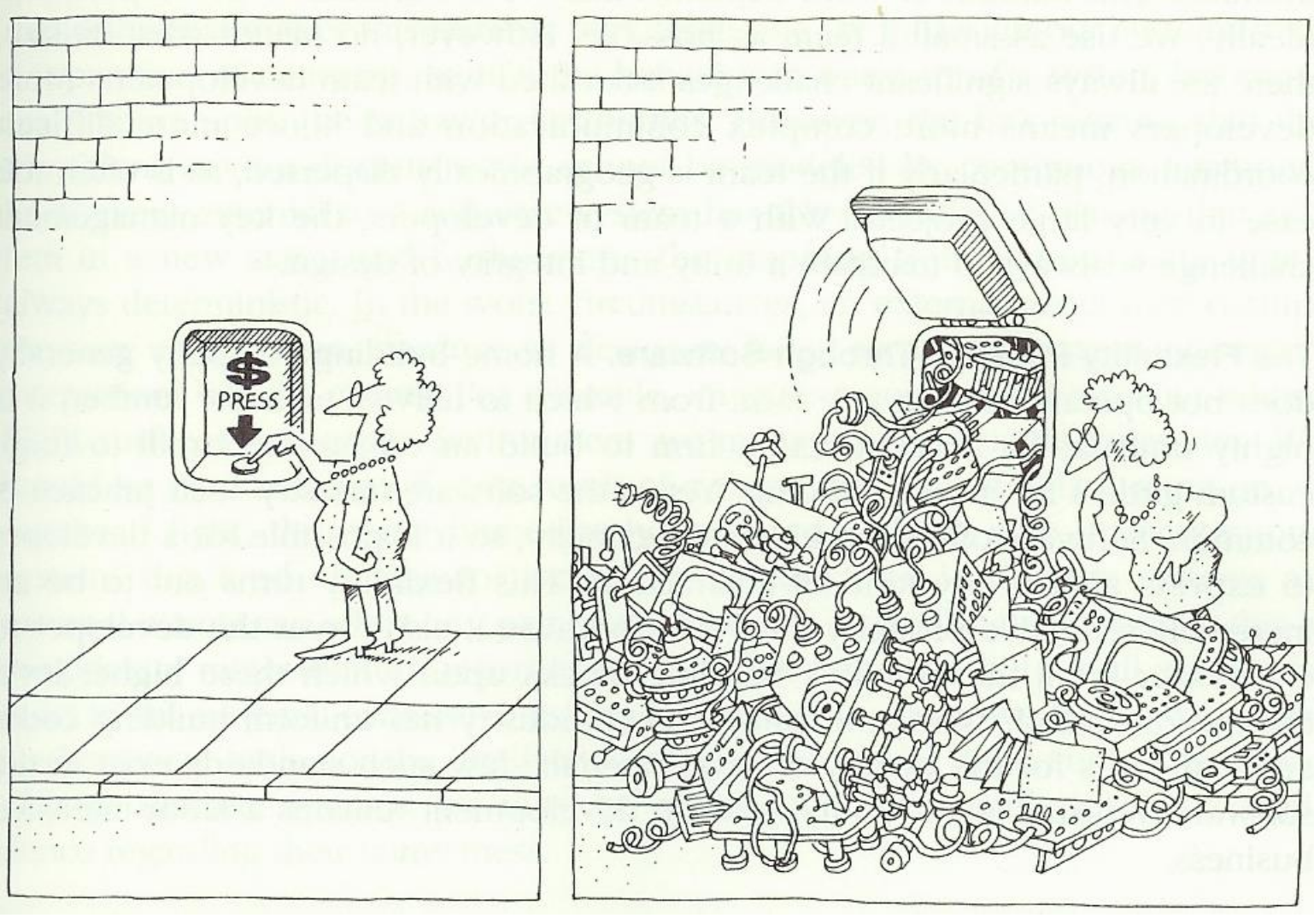
  "… es una característica inherente, por lo tanto debemos tratar de administrar la complejidad"

La Crisis del Software

- No terminar los proyectos a tiempo.
- Consumir más presupuesto del planificado.
- Baja productividad.
- Hacer productos de baja calidad.
- Gran cantidad de personal especializado dedicado a labores de mantenimiento.
- Usuarios insatisfechos con los sistemas y con los departamentos o grupos de desarrollo. Se preguntan:

  ¿Por qué el desarrollo de software es tan costoso?
  ¿Por qué toma tanto tiempo?
  ¿Hay alguna perspectiva de mejora?

La Complejidad del Software



La Complejidad del Software

4 factores que influyen en la complejidad:

✓Complejidad del dominio del problema:

- Problemas que involucran elementos de gran complejidad.
- Existencia de requerimientos contradictorios.
- Distancia entre el usuario y el desarrollador (diferentes perspectivas del problema).

✓Dificultad de administrar el proceso de desarrollo:

- Dar la ilusión de simplicidad.
- Manejo de un grupo de trabajo (problemas de comunicación, coordinación e integración).

La Complejidad del Software

- ✓ Flexibilidad exigida al software:
  - Uso estándares

- ✓ Mantenimiento del software:
  - Correctivo – corregir errores en programas (20 %).
  - Adaptativo – cambios en los requerimientos (25 %).
  - Perfectivo – tratar de mantener el software operativo (55 %).

Llegada al Caos

A medida que aumentó la complejidad del software requerido, se redujo las habilidades para manejar la complejidad.

➔ CAOS

Solución de la Ingeniería de Software:

- Tratar el software como una ingeniería.
- Uso de metodologías estructuradas.
- Mayor participación del usuario.
- Pruebas planificadas y documentadas.
- Uso de herramientas automatizadas.

Llegada al Caos

Un médico, un ingeniero civil y un profesional de la computación estaban discutiendo sobre cuál era la profesión más antigua del mundo:

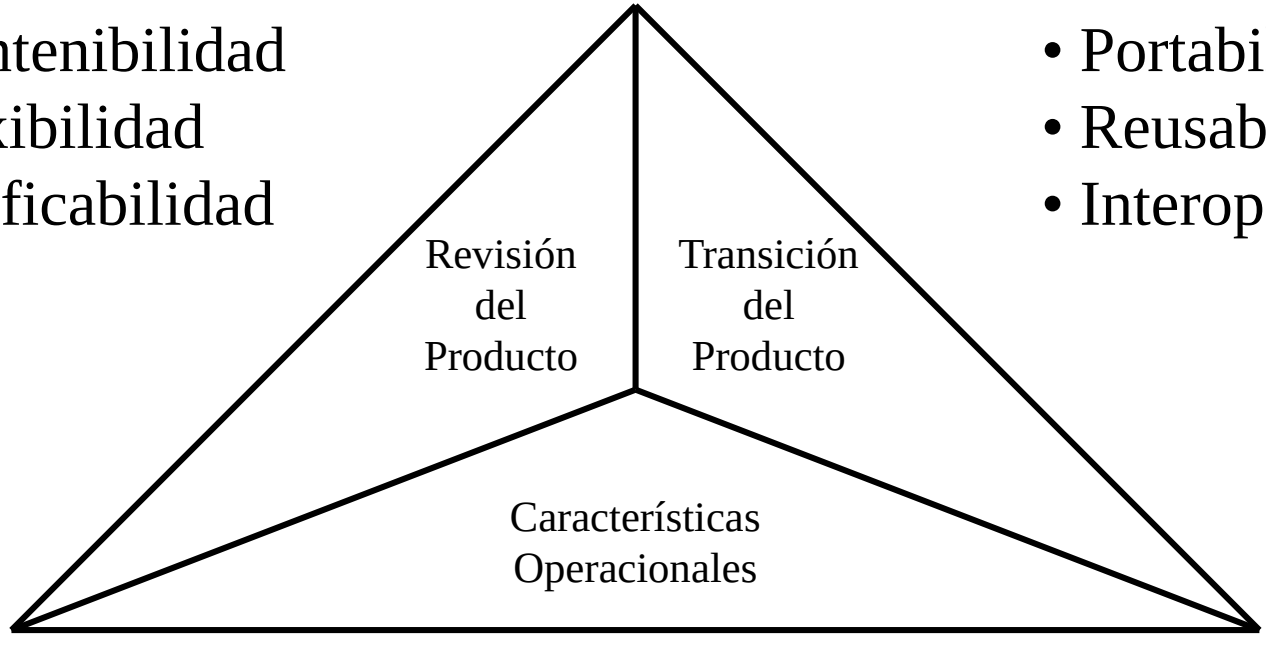
El médico: En la Biblia dice que Dios creó a Eva de la costilla de Adán, por lo tanto la mía es la profesión más antigua.

El ingeniero civil: Pero en el génesis dice que Dios impuso el orden y sacó a la tierra del CAOS en que se encontraba en siete días; esta fue la primera y más espectacular aplicación de ingeniería civil.

El profesional de la computación: Pero, ¿quién creen ustedes que creó el CAOS?

Características ideales del Software (McCall)

- Mantenibilidad
- Flexibilidad
- Verificabilidad



- Portabilidad
- Reusabilidad
- Interoperabilidad

- Corrección
- Fiabilidad
- Eficiencia

- Integridad
- Usabilidad

Características ideales del Software (McCall)

✓ Revisión del producto:

- Mantenibilidad: fácil corregir errores.
- Flexibilidad: fácil de modificar.
- Verificabilidad: fácil de probar.

✓ Transición del producto:

- Portabilidad: fácil de transportar entre ambientes.
- Reusabilidad: grado en que el todo o alguna de las partes se pueda usar en otros desarrollos.
- Interoperabilidad: fácil de acoplar.

Características ideales del Software (McCall)

✓ Características operacionales:

- Corrección: grado de satisfacción de requerimientos.
- Fiabilidad: grado de funcionamiento esperado.
- Eficiencia: grado de minimización de recursos.
- Integridad: grado de seguridad.
- Usabilidad: facilidad de aprendizaje y operación.

✓ Otros aspectos:

- Comprensibilidad: grado de poderse leer y comprender.
- Robustez: capacidad de funcionar en condiciones anormales.
- Extensibilidad: facilidad de crecer o mejorar.
- Modularidad: grado de independencia funcional de cada uno de los componentes.

3 Puntos para traer orden al Caos

✓ Descomposición:

- Descomponer el software en partes pequeñas para ser refinadas de forma independiente.
- La complejidad del problema se restringe a la complejidad de cada una de las partes.
- Es necesario hacer una división inteligente.

Conceptos relacionados

- Modularidad
- Programación estructurada

3 Puntos para traer orden al Caos

✓ Abstracción:

- Ignorar los detalles no significativos de cada elemento y trabajar con modelos ideales de éstos.
- Brinda enorme poder para manejar la complejidad.
- Ayuda a vencer las limitaciones de nuestra memoria intermedia para el manejo de la información.

Conceptos relacionados:

- Tipo Abstracto de Datos
- Conceptualización del Dominio

3 Puntos para traer orden al Caos

✓ Jerarquía:

- Organizar los elementos en niveles de categoría.
- Relaciones estructurales y semánticas.
- Incrementa el contenido semántico de las piezas de información.
- Ayuda a la comprensión del funcionamiento del sistema.

Conceptos relacionados:

- Clasificación / Generalización / tipo de
- Composición / Agregación / parte de

Orientación a Objetos

Fórmula conceptual 1:

Orientación a Objetos = Descomposición + Abstracción + Jerarquía

- Descomposición — ¿Cómo hacer que el conjunto de elementos resuelvan el problema?
- Abstracción — ¿Cómo identificar los elementos basado en los conceptos del problema?
- Jerarquía — ¿Cómo relacionar los elementos para reducir la complejidad de cada uno de ellos?

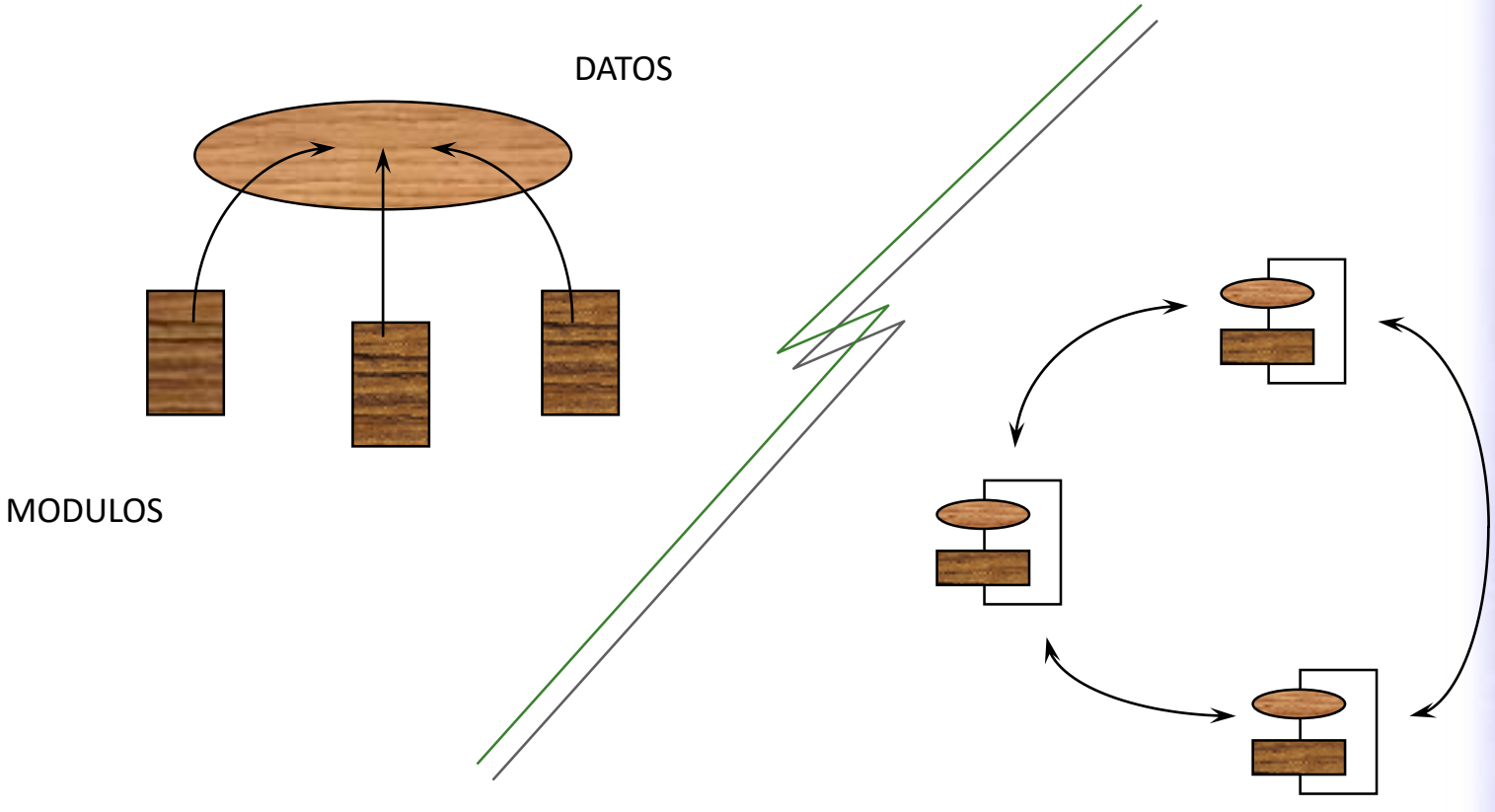
Orientación a Objetos

- Surge como una respuesta hacia la crisis del software.
- Es una nueva forma de pensar usando conceptos del mundo real.
- Significa que el software se organiza como un conjunto de objetos discretos cada uno de los cuales incorpora su estructura de datos y su comportamiento.
- “Ejecutar un programa o sistema, es algo tan sencillo como crear objetos y disparar mensajes” (A. Goldberg)
- Término introducido en el léxico con la llegada del lenguaje de programación Smalltalk.
- Conceptos a manejar: clase, objeto, método, mensaje, subclase, superclase, instancia, herencia, encapsulamiento, polimorfismo, interfaz.

Orientación a Objetos

Descomposición funcional	Orientación a Objetos
Módulos construidos alrededor de las **operaciones**	Módulos construidos alrededor de las **clases**
Datos globales o distribuidos entre los módulos	Clases débilmente acopladas y sin datos globales
Entrada / Proceso / Salida	Encapsulamiento / Mensajes
Organigramas de flujos de datos	Diagramas jerárquicos de clases

Orientación a Objetos



Orientación a Objetos

Lea la especificación de un software, que usted desee construir. Subraye los **verbos** si desea construir código procedimental, y los **nombres** si desea hacer un programa orientado a objetos

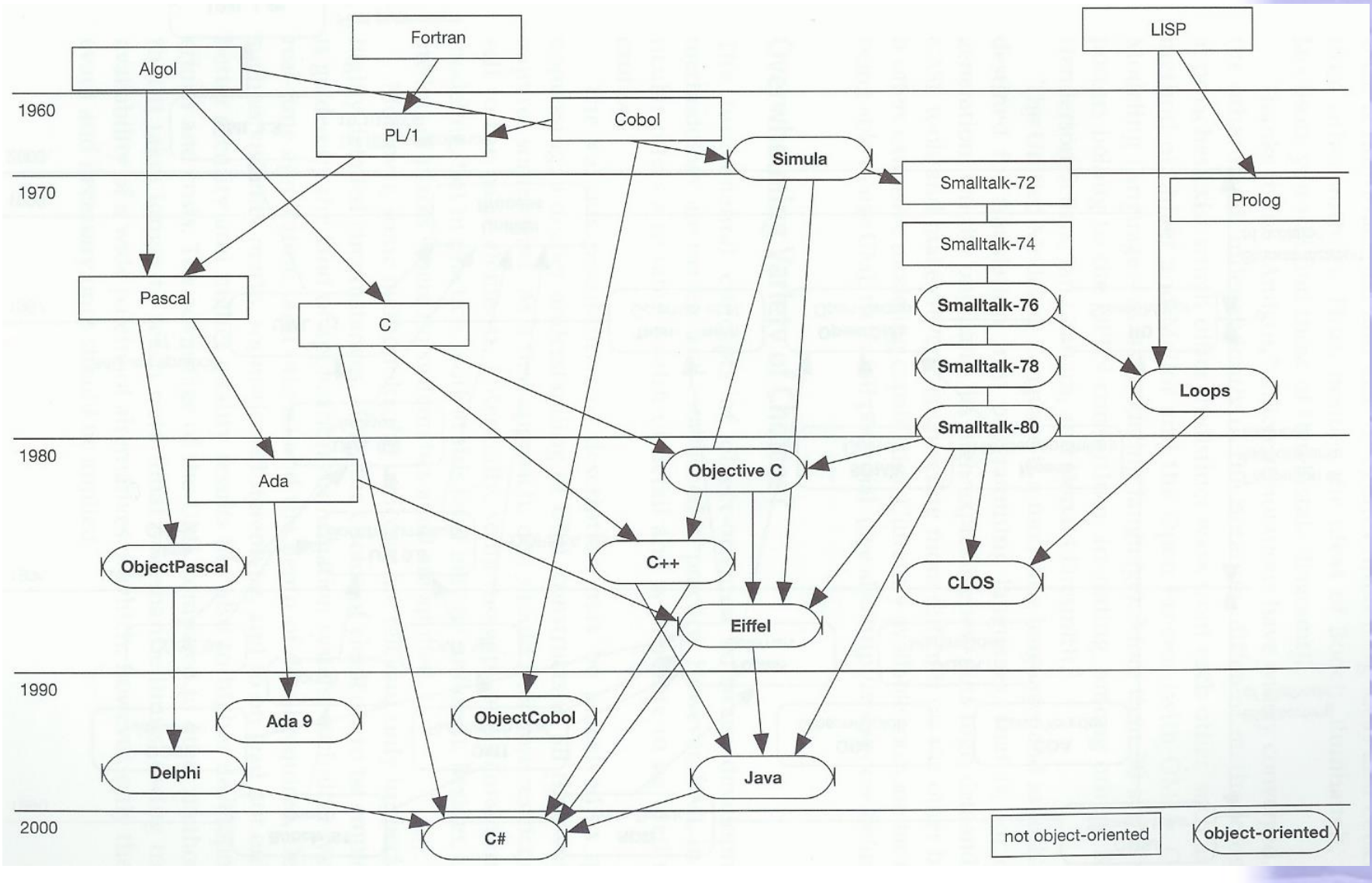
Booch 1989

Orientación a Objetos

Beneficios:

- Desarrollos más rápidos – basado en la reusabilidad de elementos.
- Calidad más alta – basado en el uso de objetos eficientes y eficaces ya existentes.
- Mantenimiento más fácil – basado en el uso de objetos libres de errores.
- Costos más bajos (programación, diseño y administración).
- Soporte a sistemas de gran escala.
- Mejores estructuras de información.
- Aumento de la adaptabilidad.

Evolución Histórica Lenguajes



Evolución Histórica Metodologías