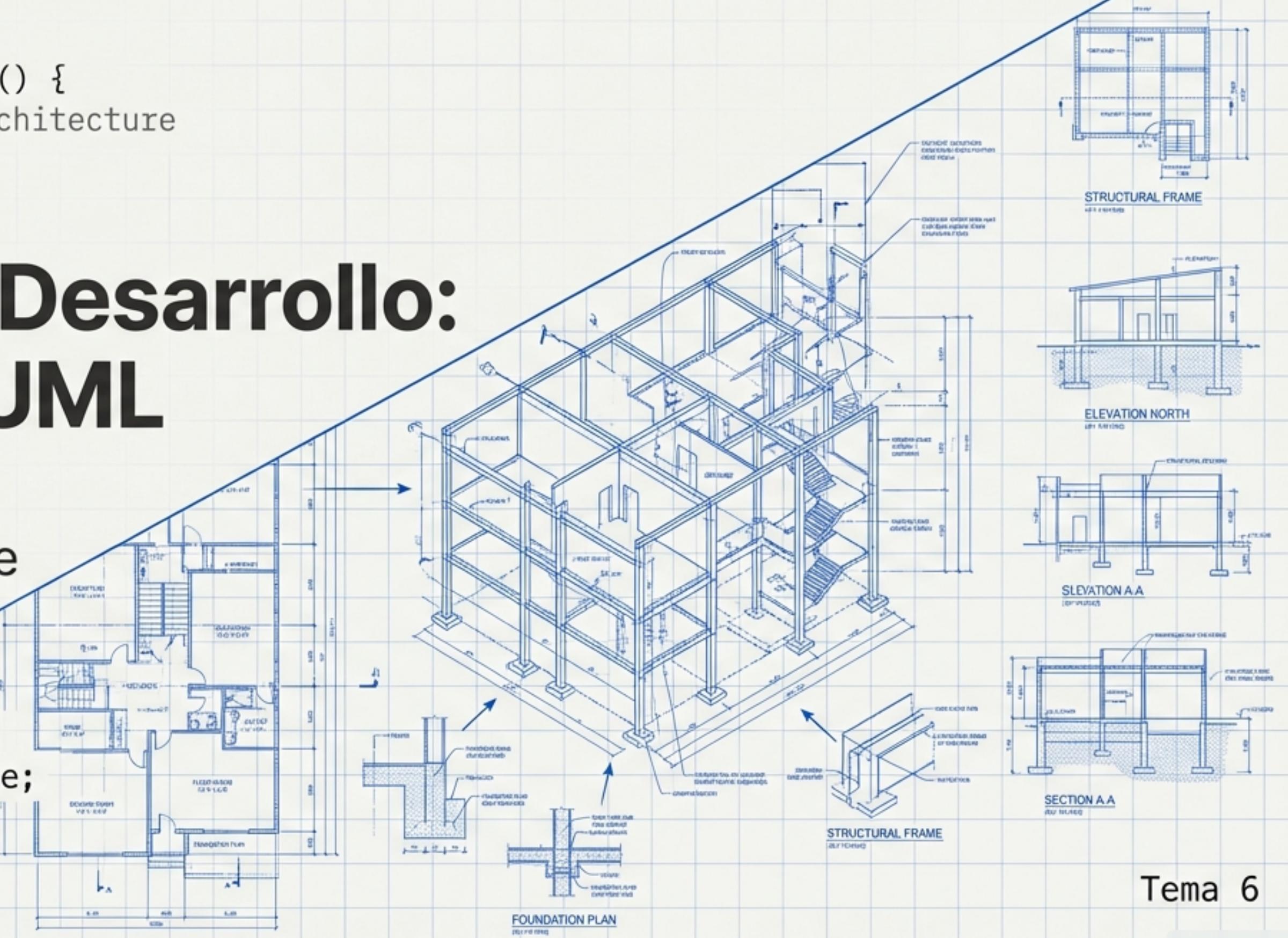


```
public class UMLDiagram implements ArchitecturalElement {  
    private String name;  
  
    public void defineStructure() {  
        // Standardizing the architecture  
    }  
}
```

# Entornos de Desarrollo: El Lenguaje UML

# Estandarización de la Arquitectura de Software

```
@Override  
public String toString() {  
    return "UML: " + this.name;  
}  
}
```





# La Necesidad del Contexto y el Modelado

## El desafío:

Modelar un sistema requiere considerar el contexto tecnológico. El enfoque cambia drásticamente entre tecnologías orientadas a objetos y las que no lo son.

## El requisito:

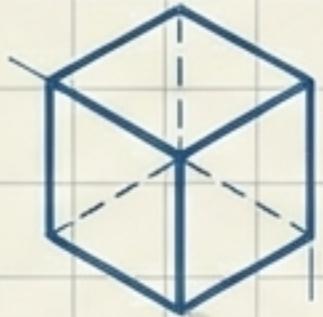
Las metodologías requieren normas y reglas estrictas para su implementación correcta.

## La analogía:

No se construye un rascacielos sin planos; no se debe construir software sin un modelo.

# Metodologías Orientadas a Objetos

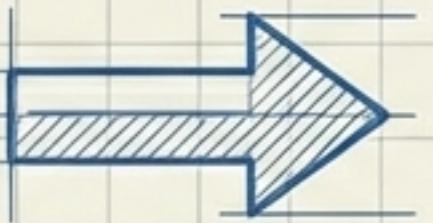
El beneficio principal de la orientación a objetos es la reutilización de componentes y la modularidad. Para lograrlo, existen metodologías predecesoras clave:



## OMT

(Técnicas de Modelado de Objetos)

Una de las primeras metodologías creadas para este paradigma.



## OPM

(Metodología de Proceso de Objetos)

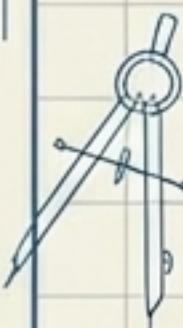
Introduce el diagrama único de proceso de objeto para modelar estructura y comportamiento.



## RUP

(Proceso Unificado de Rational)

Una metodología compleja que estructura el desarrollo en fases e iteraciones.



# La Solución Estándar: ¿Qué es UML?

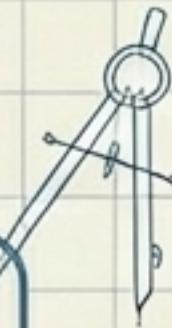
**Estándar:** Un lenguaje de propósito general para la ingeniería de software.

**Visual:** Diseñado para la visualización del diseño de un sistema.

## Unified Modeling Language (Lenguaje Unificado de Modelado)

**Versátil:** No está vinculado a un contexto de desarrollo específico, siempre que sea orientado a objetos.

**Puente:** Conecta las etapas de Análisis, Diseño e Implementación.

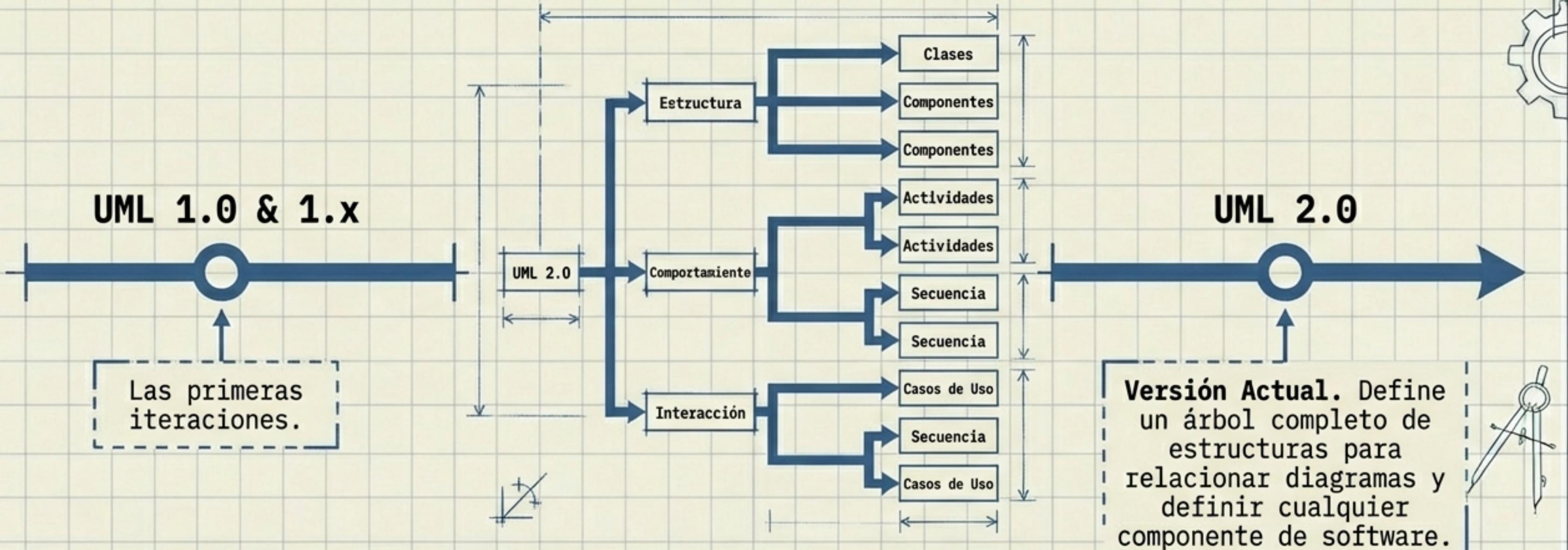


# El Valor de la Estandarización

- **Intercambio Global:** Al tener reglas definidas, los diagramas se pueden intercambiar entre diferentes regiones y equipos.
- **Reducción de Costes:** Visualizar el sistema antes de codificar reduce el coste del proceso de desarrollo.
- **Eficiencia:** Reduce el tiempo de desarrollo y solventa errores causados por la falta de documentación histórica.
- **Comunicación:** Facilita la integración de nuevos desarrolladores y mejora el entendimiento con los clientes.



# Evolución del Lenguaje

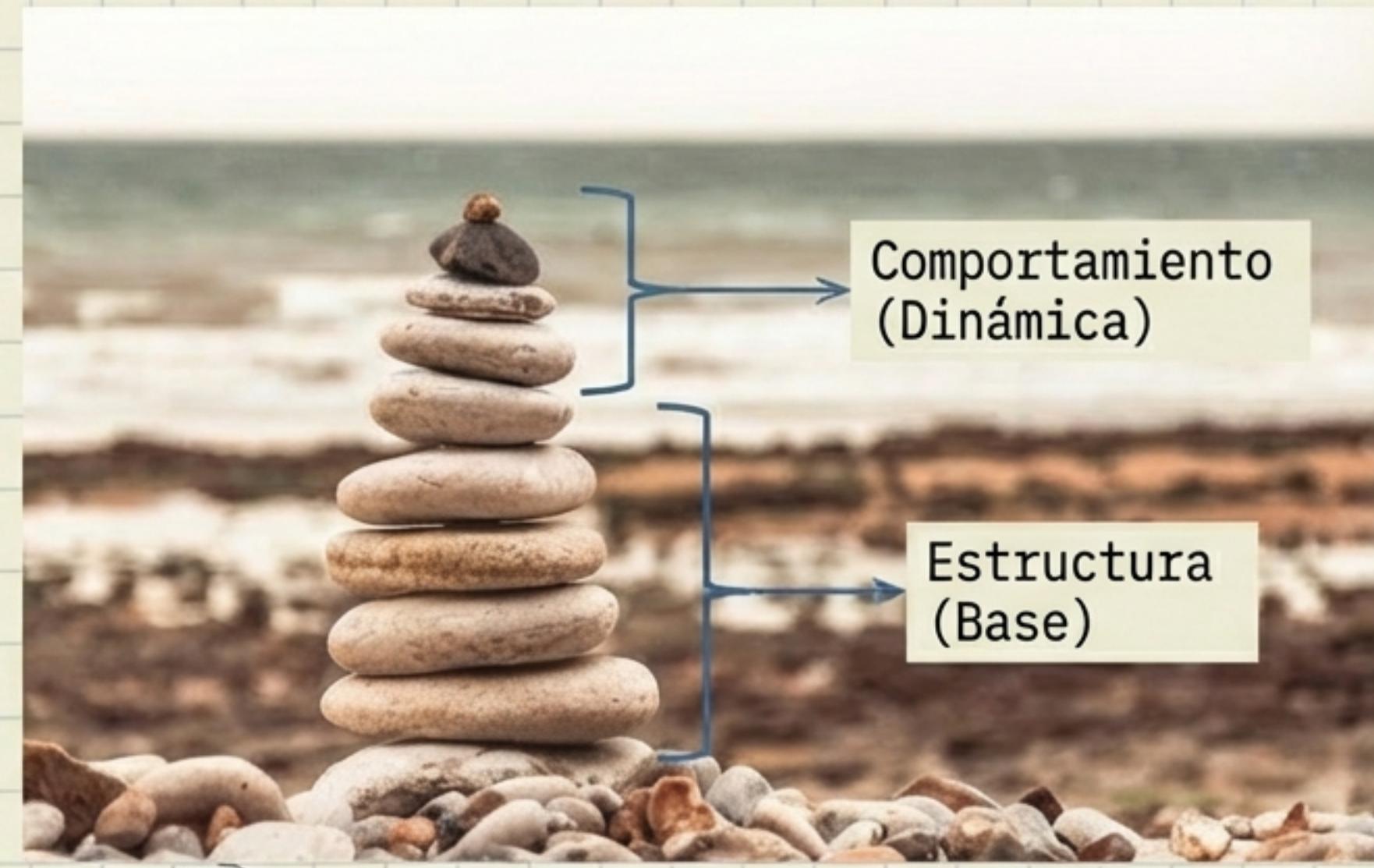


**Nota Crítica:** A pesar de ser un estándar, las reglas a veces carecen de la rigidez necesaria, lo que puede llevar a problemas de interpretación de los diagramas.

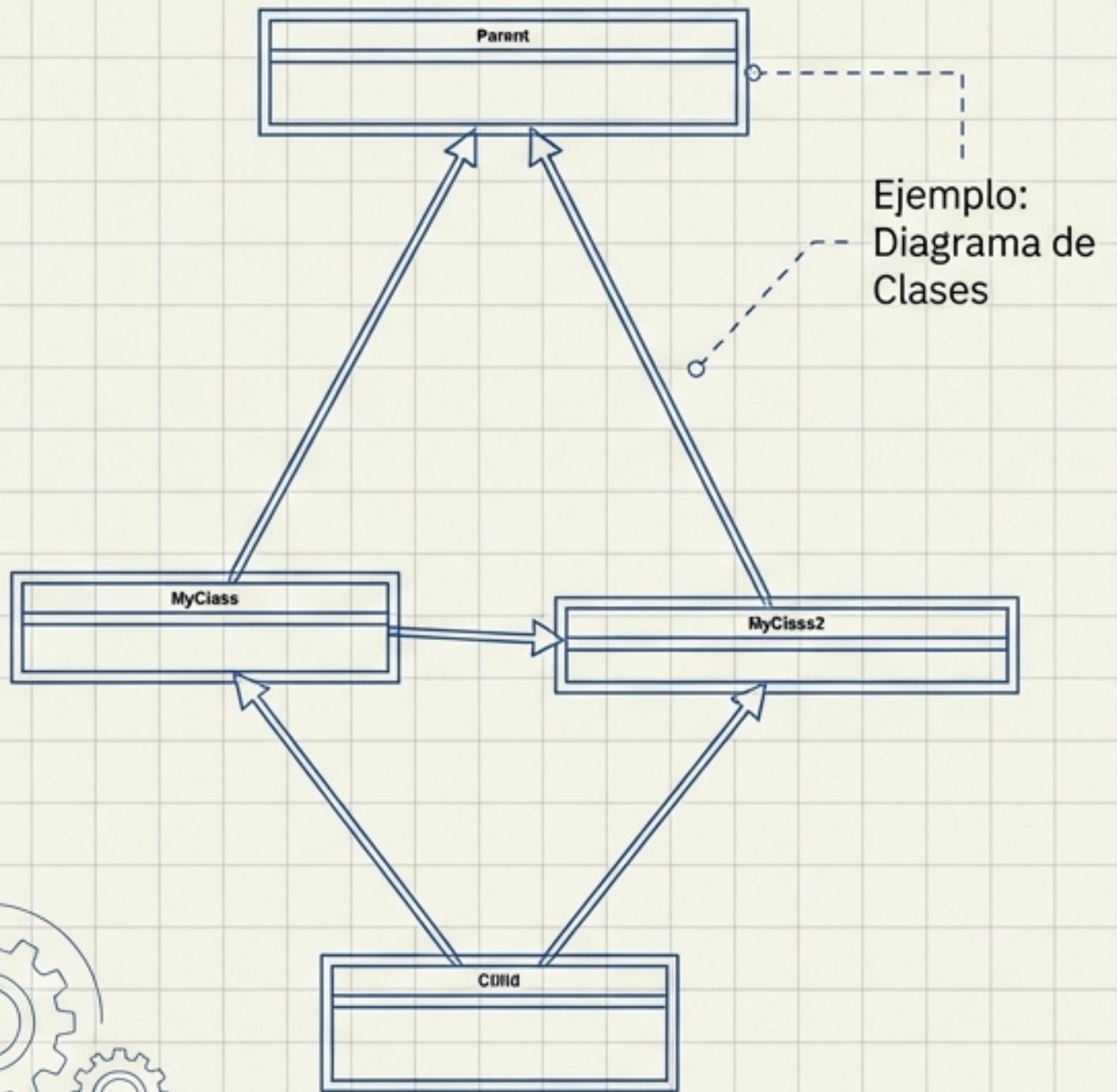
# La Colección de Diagramas

Un solo mapa no es suficiente. UML utiliza diferentes diagramas para representar distintas perspectivas y fases del desarrollo.

- **Estructura:** La anatomía estática del sistema.
- **Comportamiento:** El sistema en movimiento e interacción.



# Diagramas Estructurales: La Anatomía



## 1. Diagrama de Clases

El más destacado. Muestra las relaciones entre clases, el modelo de datos y los paquetes.

## 2. Diagrama de Componentes

Representa una visión de alto nivel de la estructura de paquetes.

## 3. Diagrama de Despliegue

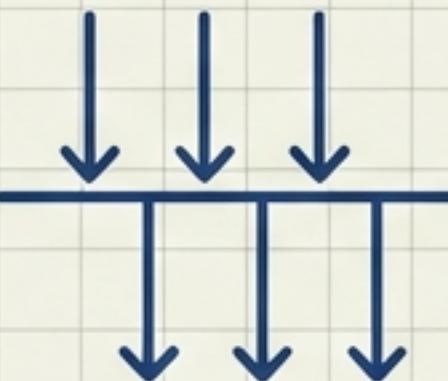
Muestra la configuración de hardware y software necesaria para la ejecución en el mundo real.

# Diagramas de Comportamiento: El Movimiento



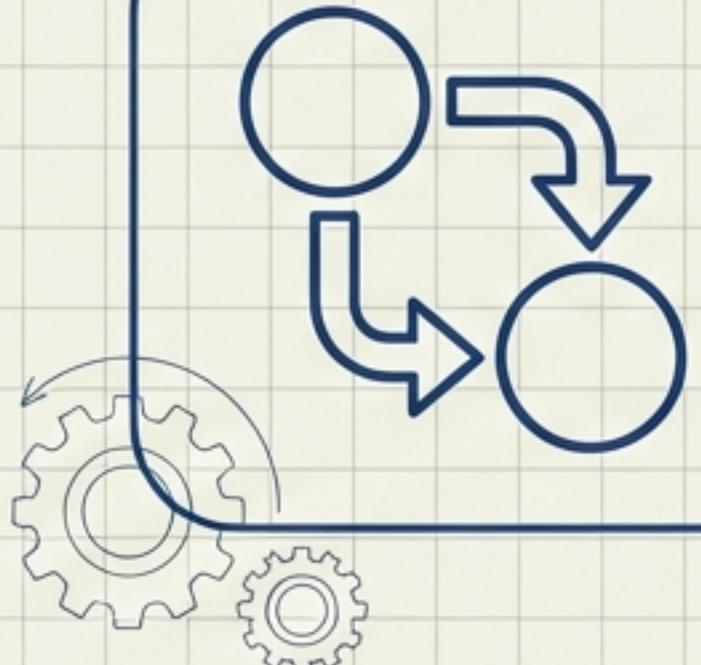
## Casos de Uso

El comportamiento del sistema desde el punto de vista del usuario.



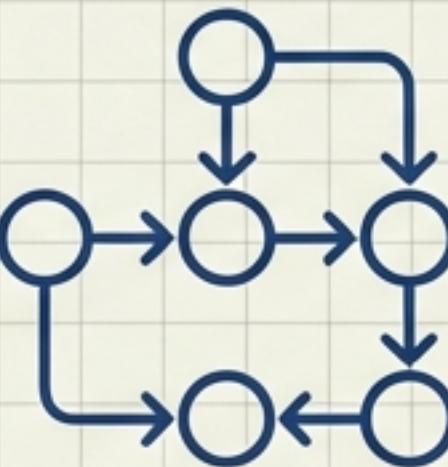
## Secuencia

Representa interacciones de manera dinámica.



## Colaboración

La interacción de un objeto con su entorno.



## Estados

La secuencia de estados y tiempo de vida de un objeto a lo largo de sus interacciones.

# Herramientas Profesionales de Modelado

Las herramientas profesionales permiten vincular el diseño con el código (IDEs).



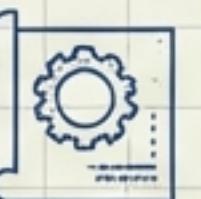
## Rational Rose

Herramienta profesional de pago, muy completa.



## StarUML

Rápida, fácil de usar y vinculable a diferentes entornos de desarrollo (IDEs).



## Papyrus UML

El estándar en el entorno Eclipse. Gratuito y de código abierto.

# Caso Práctico: La Trampa del Coste Cero

## Scenario Card

**Situación:** Un jefe propone usar una metodología nueva y desconocida solo porque sus herramientas son gratuitas.



**Veredicto:** Error. El desarrollo debe usar metodologías probadas para ahorrar costes reales y aumentar productividad.

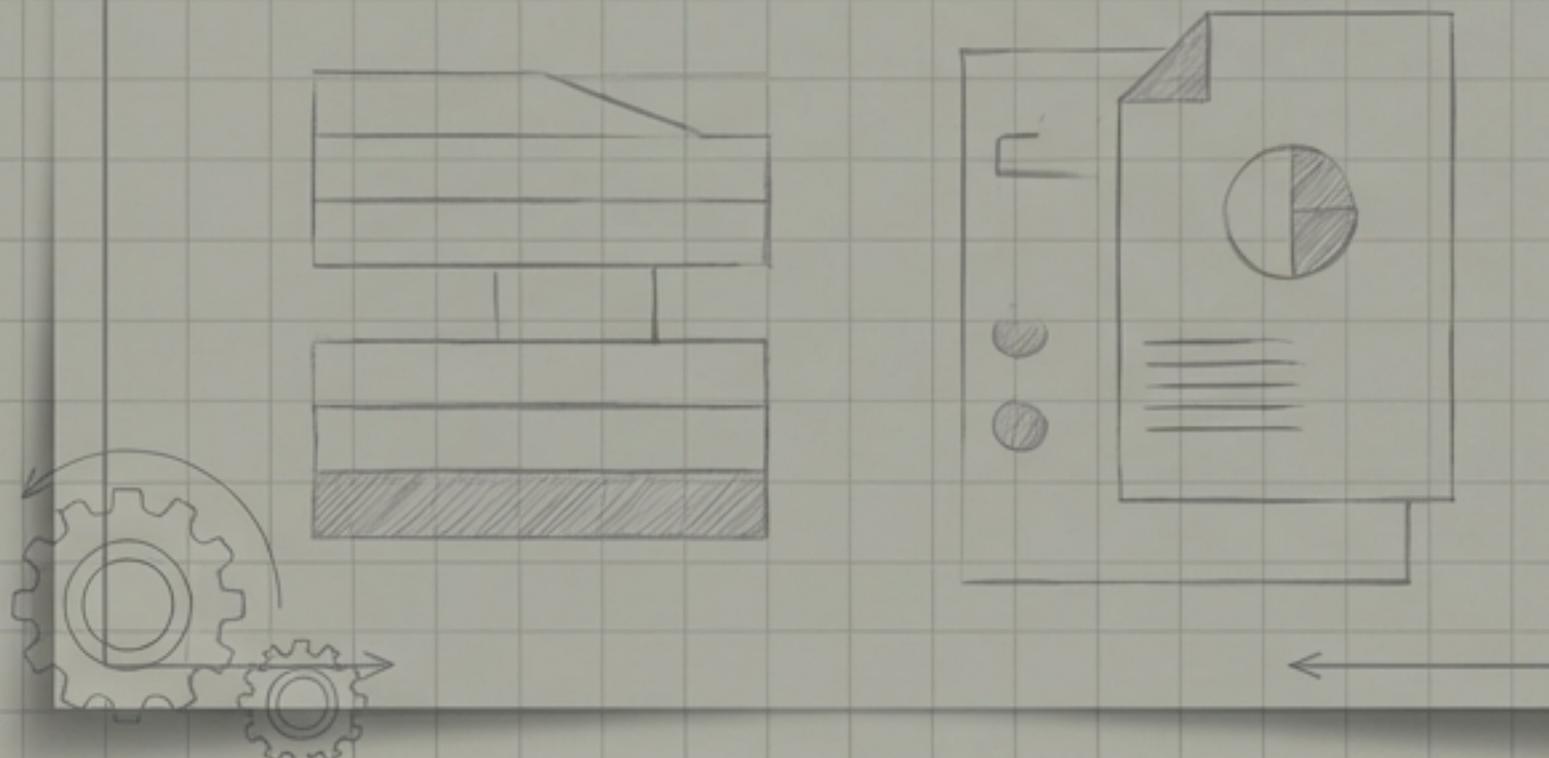


**La Lección:** Existen herramientas gratuitas para metodologías estándar (como Papyrus para UML). Elegir el estándar garantiza seguridad e intercambio de modelos validado.

# Caso Práctico: Dibujo vs. Ingeniería

## Herramientas de Dibujo (Office/Paint)

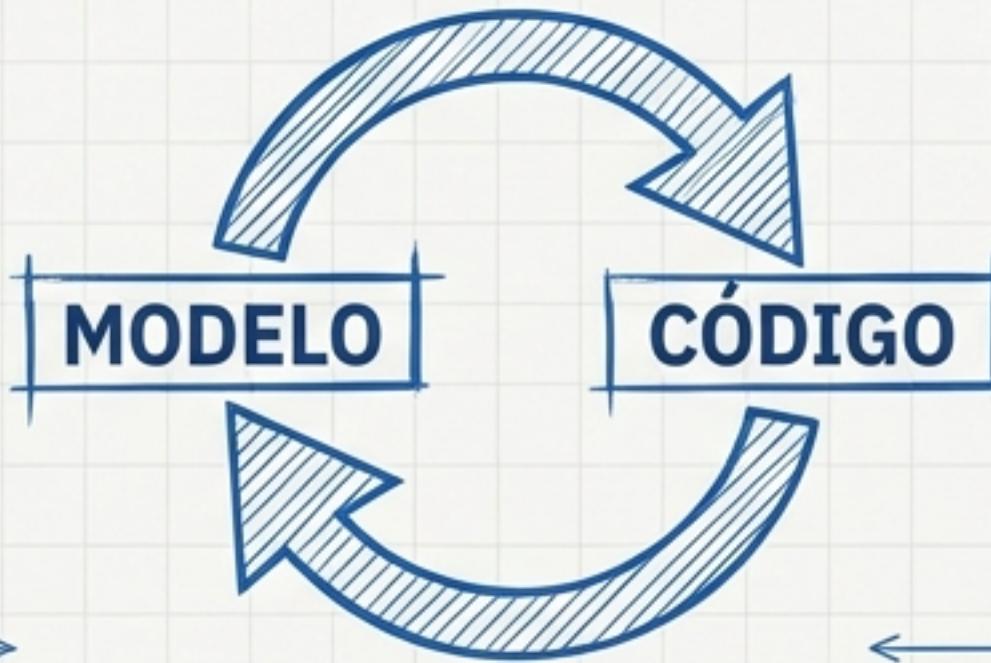
Crea documentación “muerta”.  
No se puede traducir a código  
automáticamente.



## Herramientas CASE (UML)

Crea sistemas vivos.

- **Generación de Código** (Del Modelo al Código)
- **Ingeniería Inversa** (Del Código al Modelo)



# Resumen y Buenas Prácticas

## LISTA DE VERIFICACIÓN TÉCNICA

**Metodología:** Utilizar siempre un enfoque Orientado a Objetos para software moderno.

**Estándar:** Usar UML para garantizar la interpretación universal de la documentación.

**Diagramas:** El Diagrama de Clases es fundamental para el modelo de datos.

**Herramientas:** Evitar herramientas genéricas; buscar soluciones que permitan la vinculación con el código.

# Objetivos Inteligentes en el Desarrollo



El uso de UML permite que el proceso de desarrollo cumpla con la filosofía SMART:

**Specific**  
(Específico)

**Measurable**  
(Medible)

**Achievable**  
(Alcanzable)

**Relevant**  
(Relevante)

**Time-bound**  
(Con límite de tiempo)

# Bibliografía y Referencias

- Juan, J. & Resusta, L. (2014). *UML práctico: aprende UML paso a paso.*
- Jacobson, I., Booch, G. & Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.* Addison Wesley.
- Larman, C. (2002). *UML y patrones: una introducción al análisis y diseño orientado a objetos.* Prentice Hall.