

1. INTRODUCCIÓN AL UML

Diseño de Software

Grado en Ingeniería Informática

Curso 2024/2025

José Varela Pet - Departamento de Electrónica y Computación

¿Qué significa UML?

- UML = **U**nified **M**odeling **L**anguage
- Lenguaje **gráfico** para visualizar, especificar, construir y documentar SW
- Proporciona una forma estándar de escribir los **planos** de un sistema
- Útil para cualquier persona *involucrada* en producción, despliegue y mantenimiento de software




Origen de UML

- Confluencia de tres métodos destacados desarrollados durante los años 90:
 - Método de **Booch** (Rational Software)
 - **OOSE** de **Jacobson** (Objectory)
 - **OMT** de **Rumbaugh** (General Electric)
- Alianza de los *tres amigos* en la compañía Rational Software para mejorar sus métodos

Rational
the software development company

Versiones de UML

- Aportaciones de Digital, HP, IBM, Microsoft, Oracle, Texas Instruments, Unisys, Andersen Consulting, Ericsson...
- Estandarización a cargo del Object Management Group 
- Versiones relevantes: 1.1 (1997), 1.5 (2003), 2.0 (2005)
- Versión **actual**: 2.5.1 (2017)
 - ISO 19505 (v 2.4.1)

¿Qué es un modelo?

- Es una **simplificación** de la realidad
- Proporciona **planos** de un sistema en mayor o menor detalle
- Un buen modelo resalta elementos de gran influencia y oculta los irrelevantes para un **nivel de abstracción** dado
- Dos **tipos** de modelos:
 - **Estructurales**: destacan la organización
 - **De comportamiento**: resaltan la dinámica



¿Por qué modelar?

- Para producir de forma rápida, consistente y **predecible**, con un uso **óptimo** de recursos, software de **calidad** duradera que cumpla su **propósito** y satisfaga unas necesidades que son cambiantes
- Modelado de SW aspira a incorporar los mismos principios que rigen en las ramas clásicas de la ingeniería

¿Cuándo modelar?

- Cuanto más grande y complejo es un sistema más difícil es comprenderlo en su totalidad
 - El modelado **reduce** el problema
 - Principio de *Divide y vencerás* enunciado por **Dijkstra**
- Todo sistema *útil* tiene **tendencia** natural a hacerse más complejo

Principios de modelado

1. La elección de **modelos** influye en cómo se acomete la solución de un problema
2. Todo modelo puede ser expresado a diferentes niveles de **abstracción**
3. Los mejores modelos están ligados a la **realidad**: las simplificaciones no deben enmascarar los detalles importantes
4. No basta un solo modelo: un sistema *no trivial* se aborda mejor a través de unos pocos **modelos casi independientes**

Construcción de software

- Enfoque **algorítmico** (tradicional):
 - Bloque de construcción: procedimiento/función
 - Difícil mantenimiento cuando cambian los requisitos y el sistema crece
- Enfoque **orientado a objetos** (moderno):
 - Bloques de construcción: clases y objetos
 - Válido para toda clase de dominios y cualquier abanico de tamaños y complejidades
- Mayoría de lenguajes, sistemas operativos y herramientas actuales son orientados a objetos

Un ejemplo sencillo

```
import java.awt.Graphics;  
class HolaMundo extends java.applet.Applet {  
    public void paint (Graphics g) {  
        g.drawString("¡Hola, Mundo!", 10, 10);  
    }  
}
```

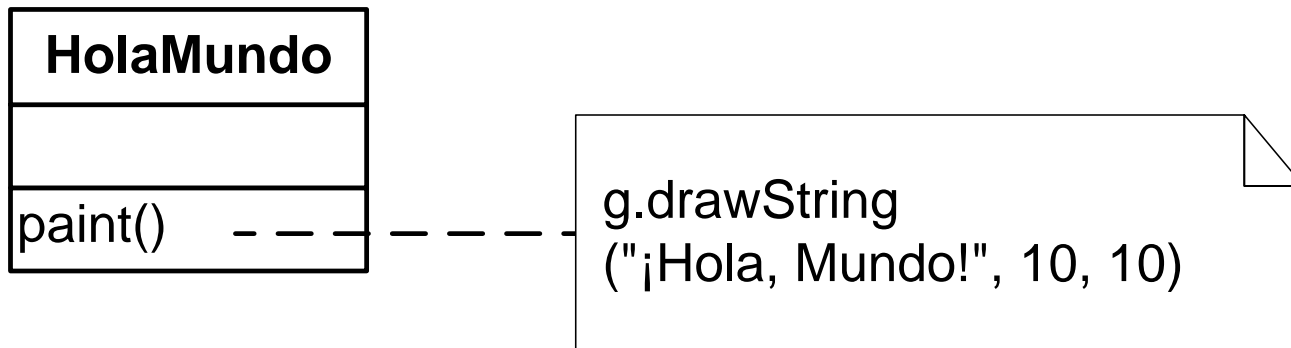
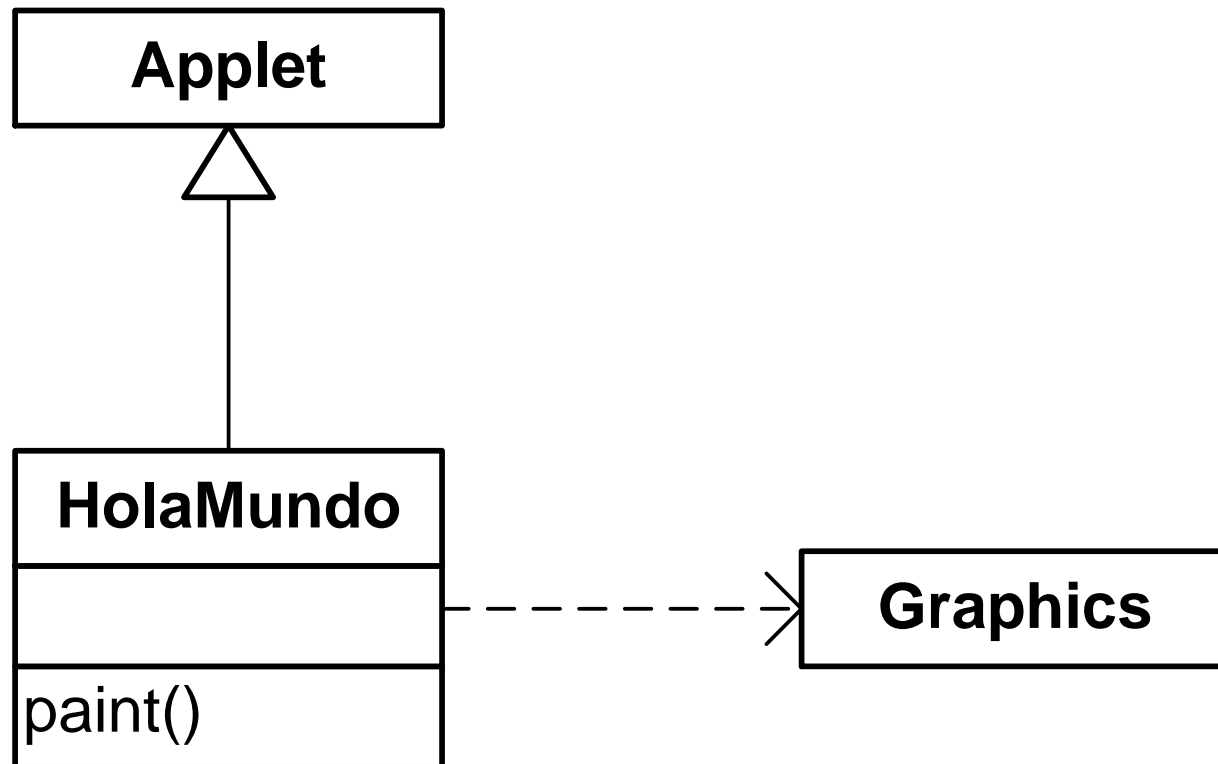
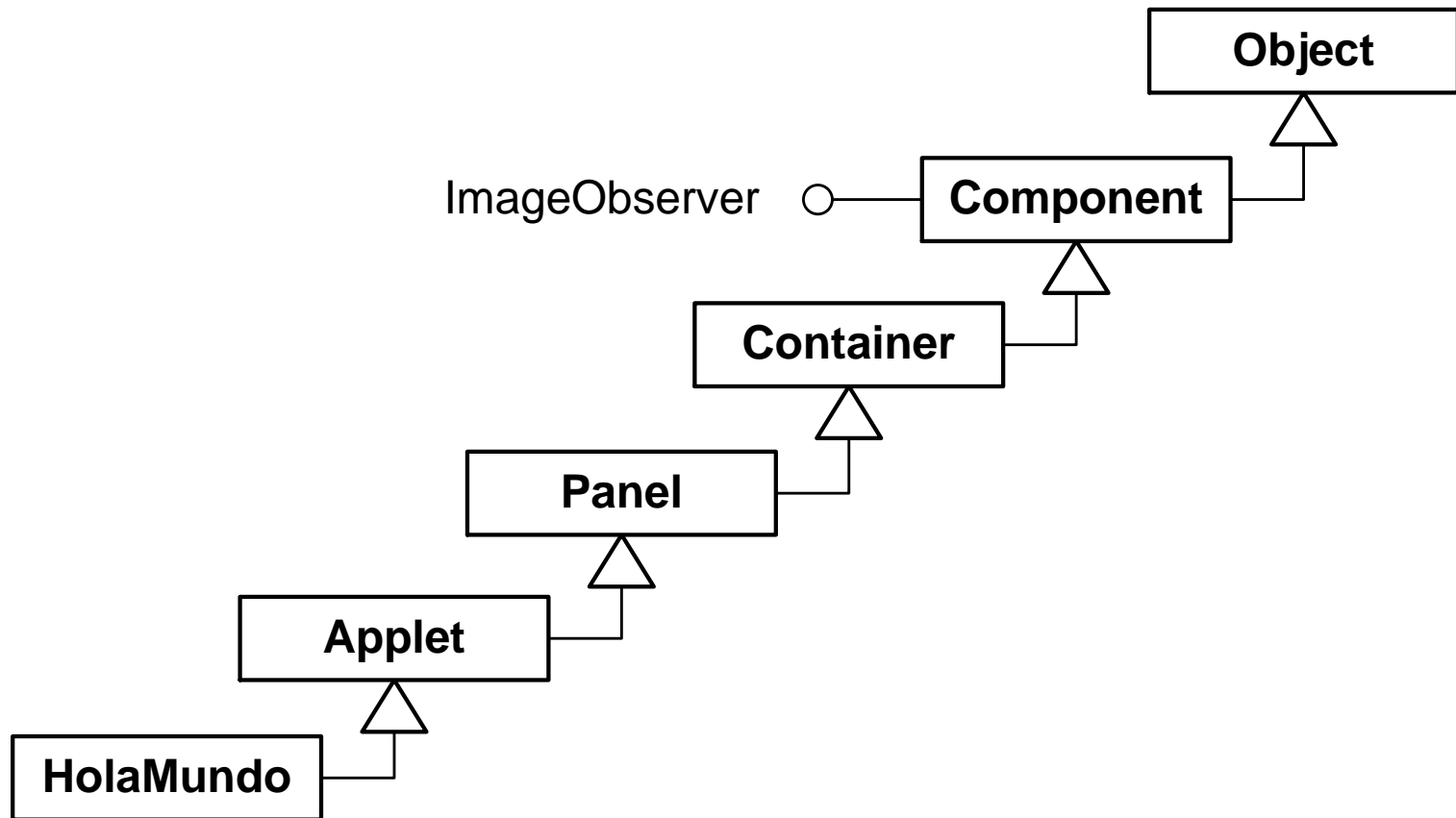


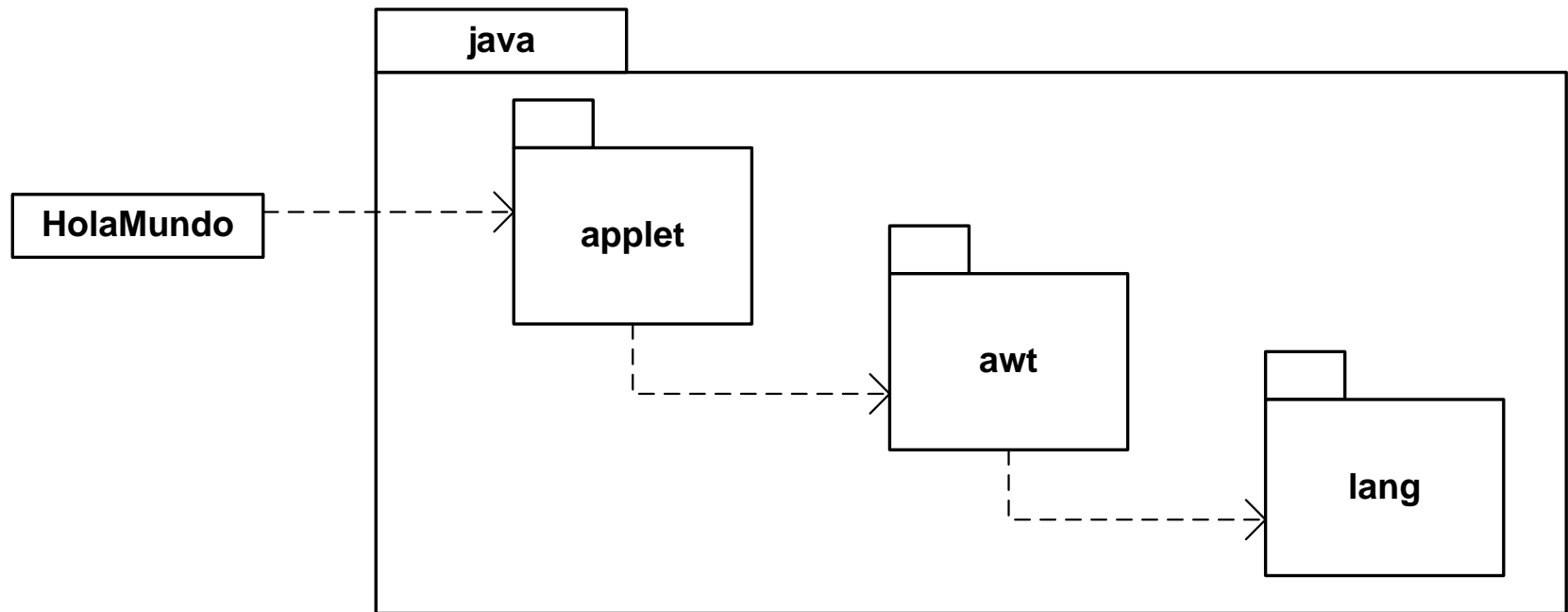
Diagrama de clases



Jerarquía de clases



Organización en paquetes



Interacción entre objetos

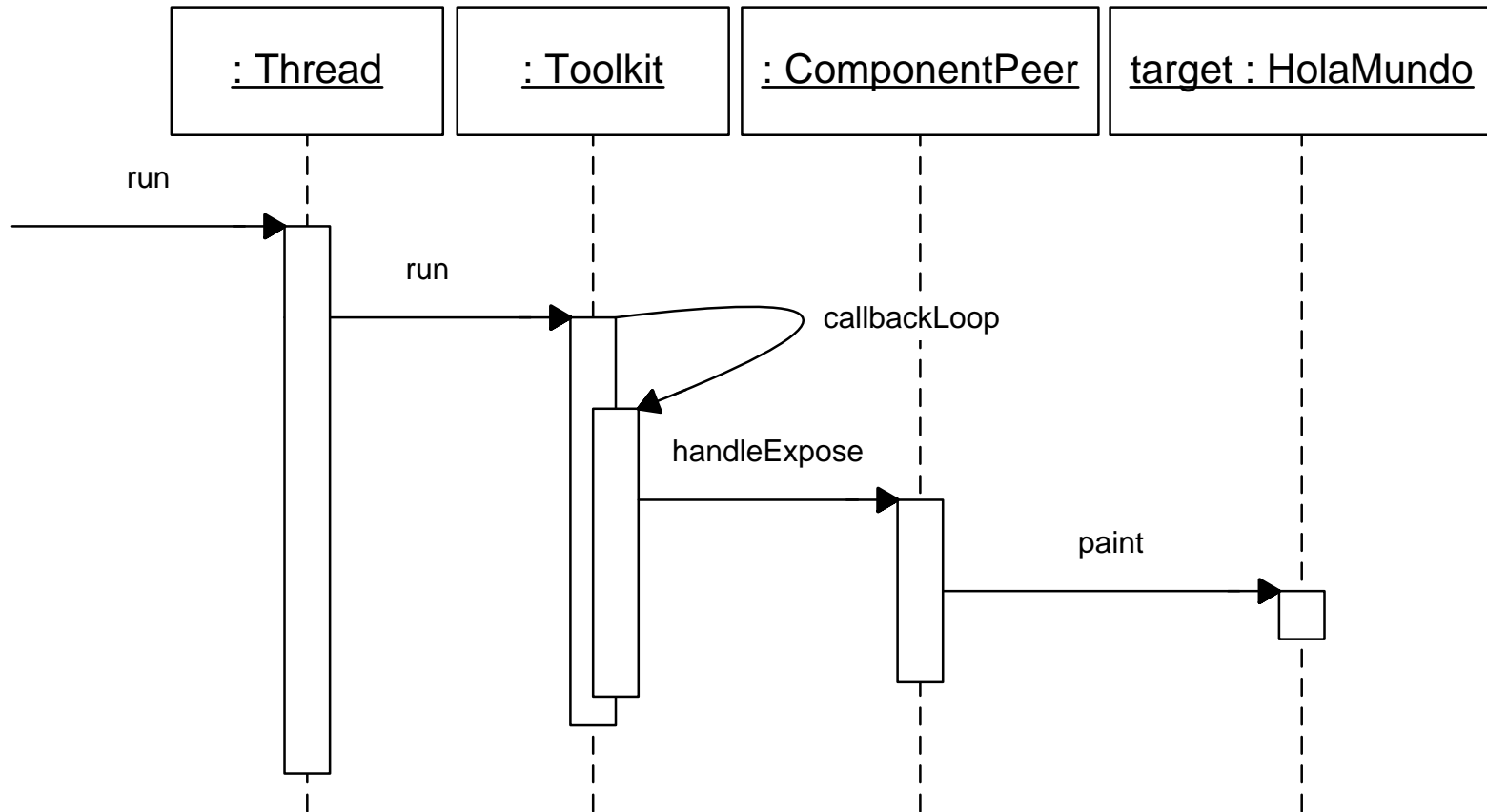
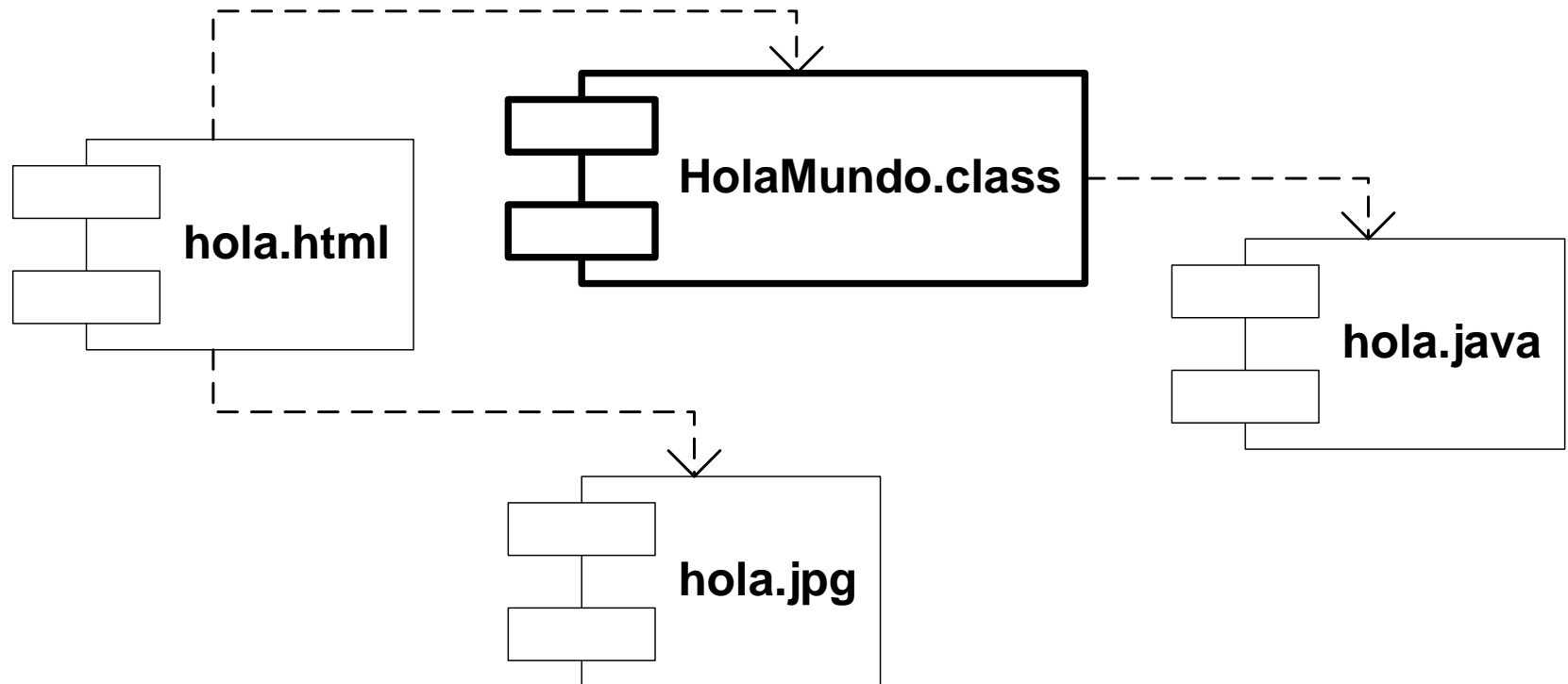


Diagrama de componentes



Visión general de UML

- UML es un **lenguaje** de modelado estándar para escribir planos de software
- Se utiliza para:
 - **Visualizar** la arquitectura del sistema
 - **Especificar** su estructura y comportamiento
 - Proporcionar plantillas para **construir**
 - **Documentar** las decisiones adoptadas
 - En fin, **comprender** mejor el sistema a desarrollar y controlar el **riesgo**

UML es un lenguaje (de modelado)

- Todo lenguaje aporta un **vocabulario** y las **reglas** para combinar esas palabras
- Un lenguaje **de modelado** indica cómo crear y leer modelos bien formados, pero no qué modelos crear ni cuándo crearlos
- El lenguaje es sólo una parte de cualquier **método** de desarrollo de software:

Método = Lenguaje + Proceso

Bloques de construcción de UML

1. **Elementos** (abstracciones)
2. **Relaciones** entre elementos
3. **Diagramas** que muestran los elementos y sus interrelaciones

Reglas de UML

- Los bloques de construcción no pueden combinarse de cualquier modo
- UML tiene reglas semánticas para generar modelos *bien formados*
- Durante el modelado es común construir modelos abreviados, incompletos o inconsistentes

Diagramas

- Con UML se forman modelos a partir de **bloques** de construcción
 - **Modelo**: abstracción o simplificación de la realidad, completa y consistente
- **Diagrama**: *grafo* conexo de *nodos* (elementos) y *arcos* (relaciones)
- Permiten visualizar un **sistema** desde **diferentes perspectivas** resumidas

Vistas de modelado

Vocabulario
Funcionalidad

Vista de
diseño

Ensamblado del sistema
Gestión de configuraciones

Vista de
implementación

Comportamiento

Vista de
casos de uso

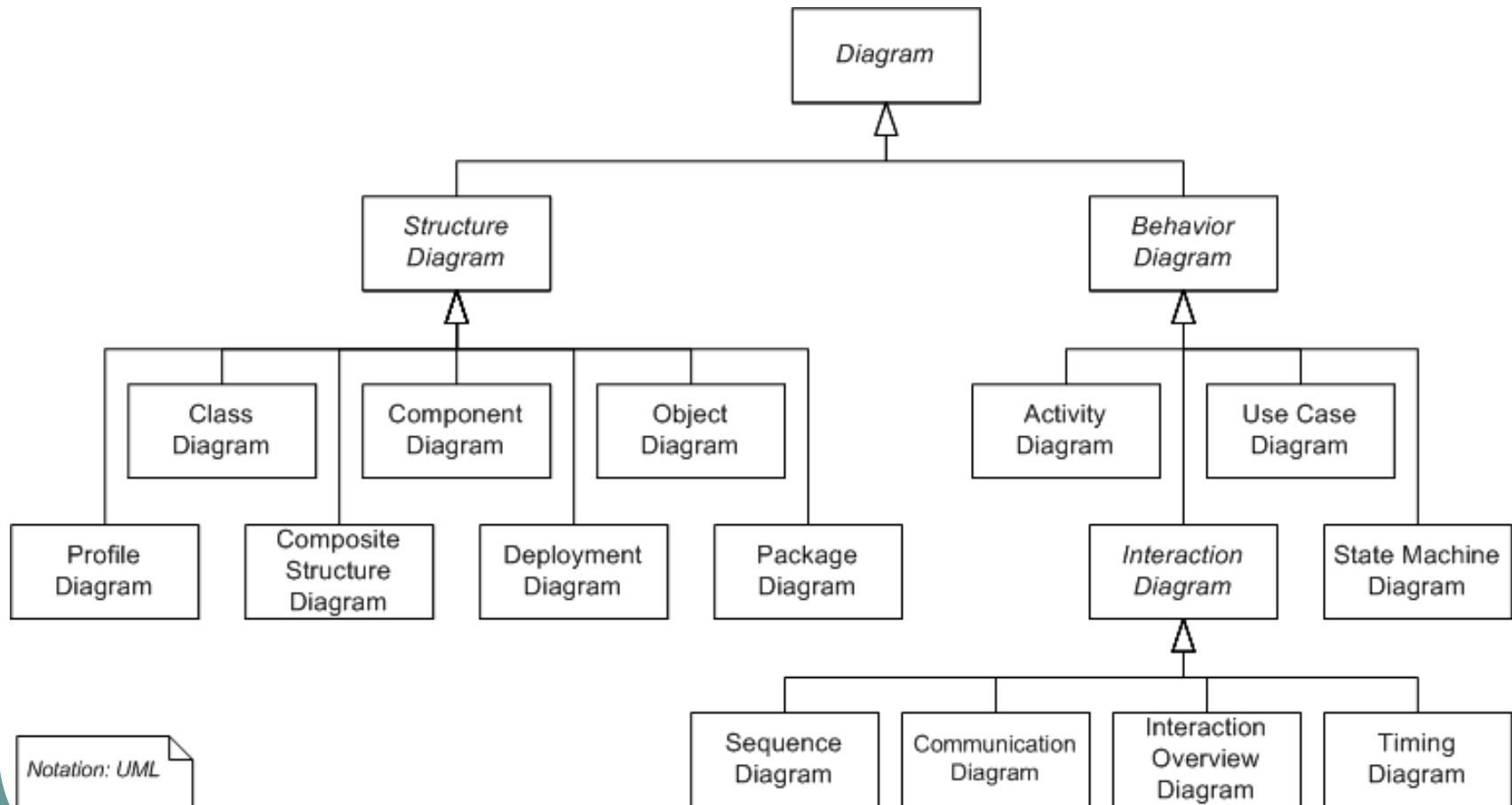
Vista de
procesos

Vista de
despliegue

Funcionamiento
Capacidad de crecimiento
Rendimiento

Topología del sistema
Distribución
Entrega e instalación

Tipos de diagramas



Tipos de diagramas en UML 1.X

- Modelado de la parte **estática** del sistema
 1. Diagramas de clases
 2. Diagramas de objetos
 3. Diagramas de componentes
 4. Diagramas de despliegue
 - Modelado de la parte **dinámica** del sistema
 5. Diagramas de casos de uso
 6. Diagramas de secuencia
 7. Diagramas de colaboración
 8. Diagramas de estados
 9. Diagramas de actividades
- } Diagramas de interacción

Cambios en UML 2.X

- Inclusión oficial de los diagramas de **paquetes**
- Los diagramas de colaboración pasan a denominarse de **comunicación**
- Nuevos tipos de diagramas:
 - Temporización (*Timing*)
 - Estructura compuesta (*Composite Structure*)
 - Vista de interacción (*Interaction Overview*)
 - Perfil (*Profile*)

Características de los diagramas

- Precisan un **nombre** único
- Se ubican en **paquetes**
- Pueden representar cualquier combinación de **elementos** de UML
- ...aunque lo habitual es que en cada uno sólo figuren elementos de unos pocos tipos

Recomendaciones al crear diagramas

- Tener presente su **propósito** y nombrarlos significativamente
- Revelar sólo el nivel de **detalle** suficiente evitando diagramas **minimalistas**
- Mantener equilibrio entre diagramas **estructurales** y de **comportamiento**
- Evitar diagramas **extraños** o **redundantes**
- No obsesionarse con el **formato** ni empeñarse en conservarlo todo

Diagrama bien estructurado

- Se centra en un **aspecto** del sistema
- Contiene sólo **elementos** y **relaciones** esenciales para ese aspecto
- Proporciona **detalles** de forma consistente con su nivel de abstracción
- Evita los **cruces** de líneas
- **Coloca** cerca los elementos que también son semánticamente próximos