

# *Qué es UML*

Análisis y Diseño Orientado a Objetos

M.Sc Ing. Irvin Vásquez



# Qué es UML?

El **L**enguaje **U**nificado de **M**odelado es un lenguaje visual para modelado de sistemas de propósito general.

Está asociado frecuentemente al modelado para software de sistemas con orientación a objetos.

Los diagramas de UML son fácilmente entendidos por las personas y son reconocidos por las computadoras para la generación de software.

---

---

# Qué es UML?

El UML **NO** nos da una clase de metodología de modelado sino que provee una sintaxis que puede ser usada para construir modelos.

El Proceso Unificado (UP) es una metodología que nos dice que empleados, actividades, y artefactos necesitamos para crear un modelo de software del sistema.

---

---

# *Qué es el UML*

- ⇒ UML no está ligado a una metodología de desarrollo o un ciclo de vida, es capaz de ser utilizado por todas las metodologías existentes.
- ⇒ UML y UP trabajando juntos unifican toda la experiencia de la ingeniería de software en la actualidad en cuanto a lenguajes de modelado visual.

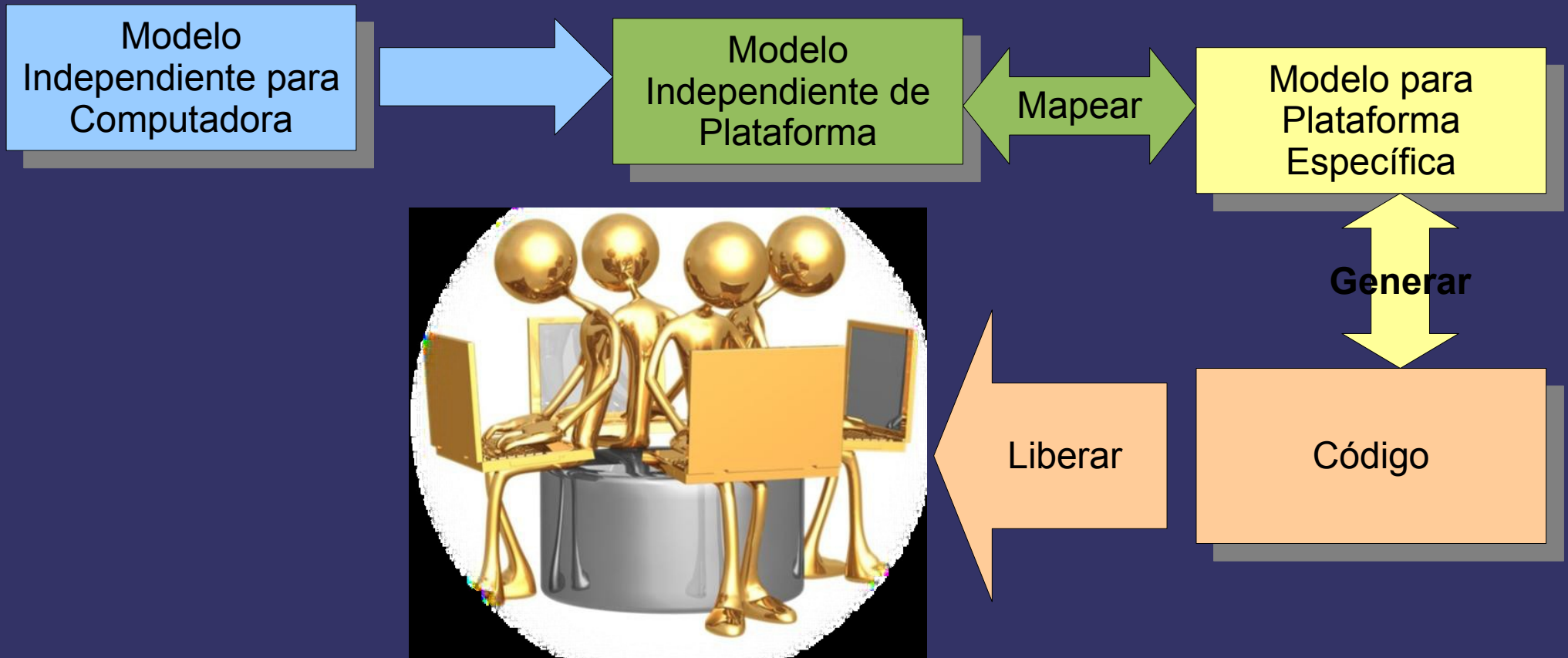
# *Nacimiento y Futuro del UML*

- ⇒ Varias iniciativas de lenguajes de modelado visual y de OO, con muchas debilidades cada uno.
  - ⇒ Se establecen con líderes Booch, Rumbaugh y Jacobson (Método Booch, Modelaje de objetos, Estudio de Casos).
  - ⇒ Se prevé que el futuro de UML es MDA que es la metodología de desarrollo basado en modelos.
- 
-

# *Model Driven Architecture (MDA)*

- ➔ MDA Define una visión para como el software puede ser desarrollado basado en modelos.
  - ➔ La esencia: El modelo puede ser migrado a producción automáticamente mediante arquitectura de software.
  - ➔ El software es producido en base a una serie de transformaciones a los modelos por una herramienta de modelado MDA.
  - ➔ Un modelo abstracto independiente de la computadora (CIM) es usado como base para un Modelo Independiente de la Plataforma (PIM)
  - ➔ PIM es transformado en un Modelo para Plataforma Específica (PSM) que es transformado en Código.
- 
-

# *Model Driven Architecture (MDA)*



# *Model Driven Architecture (MDA)*

## ⇒ El CIM (Modelo):

- Alto nivel de abstracción
- Captura todos los requerimientos claves del sistema
- Captura el vocabulario del dominio del problema
- Independiente de las computadoras.

## ⇒ PIM (Modelo):

- Expresa la semántica de los procesos de negocio para el sistema de software
- Independiente de la plataforma
- Mismo nivel de abstracción que el modelo de análisis
- Más completo que el modelo de análisis

## ⇒ Ejemplos de MDA.

- iUML de Kennedy Carter provee Action Specification Language
  - ArcStyler, 70 a 90% de artefactos son creados, pero el cuerpo de operación debe ser completado por el usuario en Java o C++
  - Open Source: revisar sitio de OMG MDA: Eclipse y AndroMda.
- 
-



# ***Por qué Unificado?***

- ➔ La unificación no solo es un evento histórico, sino que se ha venido unificando a través de varios dominios:
    - Ciclo de Vida de Desarrollo: UML provee una sintaxis visual para el modelado a través del ciclo de vida de desarrollo de software, desde la ingeniería de requerimientos hasta la implementación.
    - Dominios de Aplicación. UML puede ser usado para modelar desde sistemas incrustados hasta sistemas de soporte a decisiones.
    - Lenguajes de Implementación y Plataformas. Es un lenguaje Neutral y de plataforma neutral. Puede ser usado con lenguajes OO puros como Smalltalk, Java y C#, o con lenguajes híbridos como C++ o basados en objetos como VB.
- 
-

# ***Por qué Unificado?***

- Desarrollo de Procesos. UP y sus variantes son probablemente el desarrollo de procesos preferido para sistemas OO. UML puede soportar (y lo hace) muchos otros software de ingeniería de procesos.
- Tiene su propio set de conceptos internos. UML valientemente trata de aplicar sus propios conceptos, pero todavía está en pleno desarrollo



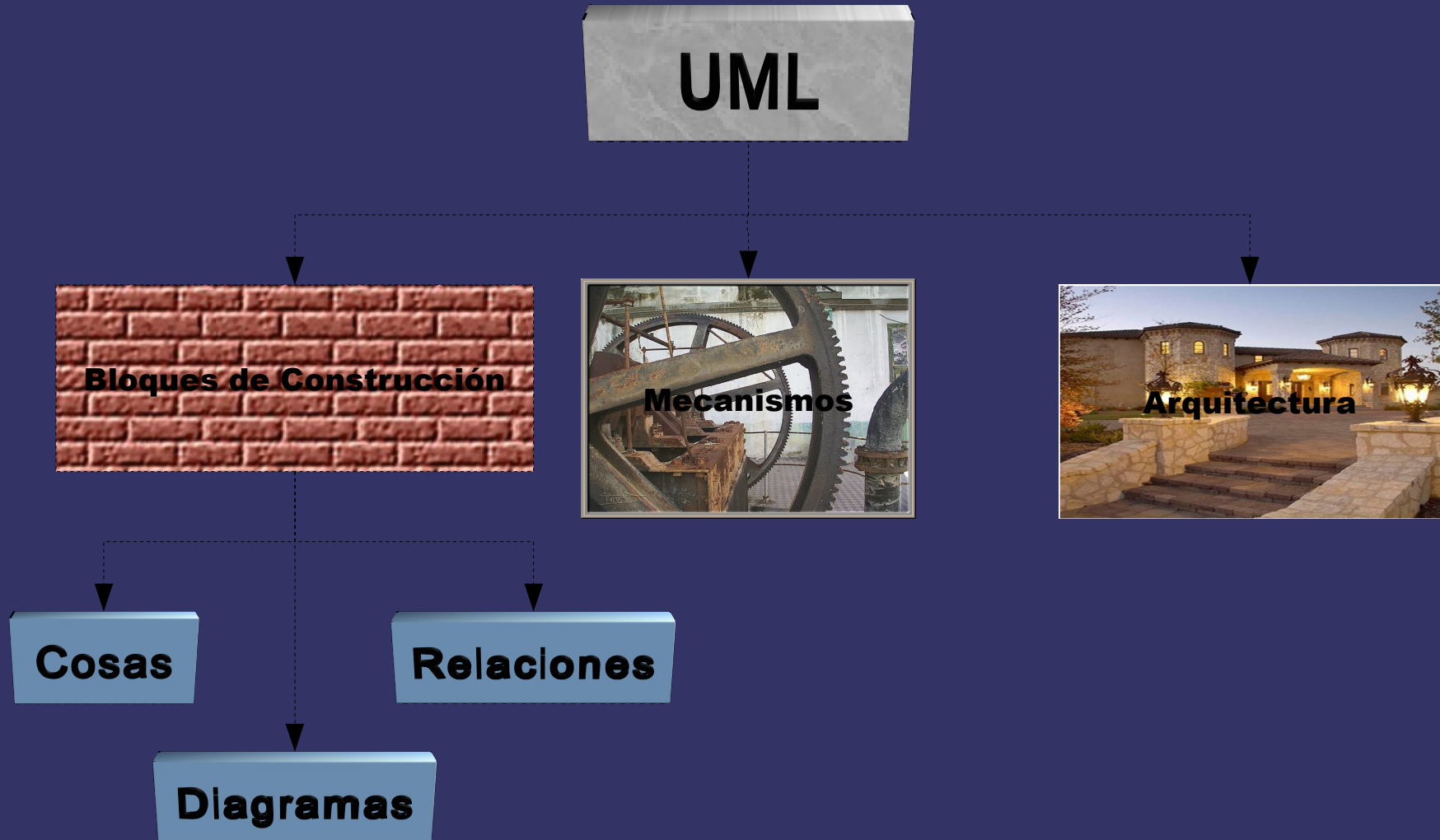
# ***Objetos y UML***

- ➔ La premisa básica de UML es que “nosotros podemos modelar software y otros sistemas como colecciones de objetos que interactúan”.
  - ➔ Existen dos aspectos a considerar en un modelo:
    - Estructura estática. Describe que objetos son importantes en el modelado y como se relacionan
    - Comportamiento dinámico: describe el ciclo de vida de esos objetos y como interactúa con cada uno de los otros para enviar los requerimientos de funcionalidad al sistema.
    - Un objeto es un segmento compacto de datos y comportamientos, en otras palabras, los objetos contienen datos y desempeñan funciones.
- 
-

# ***Estructura UML***

- ⇒ La forma de entender el UML es entendiendo su estructura:
    - Bloques de construcción: son los elementos de modelado básico de UML, relaciones y diagramas.
    - Mecanismos comunes: Las formas comunes que UML utiliza para alcanzar una meta.
    - Arquitectura. La visión de la arquitectura de UML.
  - ⇒ UML también fue modelado y diseñado usando UML. Este diseño es el **metamodelo** de UML
- 
-

# ***Diagrama de Estructura UML***










# ***Bloques de Construcción de UML***

⇒ UML está compuesto por 3 bloques de construcción:

- **Cosas:** Los elementos que se están modelando, que se pueden dividir en:
  - Cosas estructurales: sustantivos, como: Clase, Interfaz, colaboración, caso de uso, clase activa, componente, nodo.
  - Cosas funcionales: verbos: interacciones, actividades, estados de máquinas.
  - Cosas agrupadas: paquetes.
  - Cosas documentales: agregadas para capturar información, estilo post-it.
- **Relaciones:** permiten mostrar como dos cosas se relacionan una con otra.
- **Diagramas:** Ventanas o Vistas dentro del modelo.

# Relaciones.

Type of relationship	UML syntax source target	Brief semantics
Dependency		The source element depends on the target element and may be affected by changes to it
Association		The description of a set of links between objects
Aggregation		The target element is a part of the source element
Composition		A strong (more constrained) form of aggregation
Containment		The source element contains the target element
Generalization		The source element is a specialization of the more general target element and may be substituted for it
Realization		The source element guarantees to carry out the contract specified by the target element

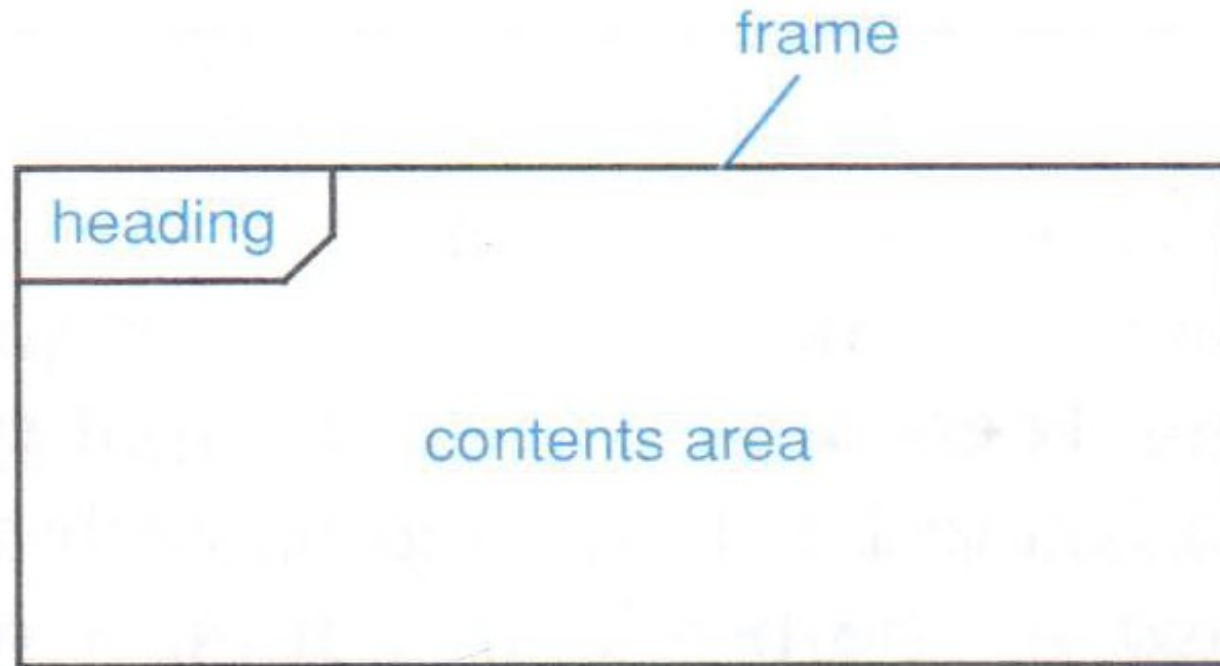


# *Diagramas*

- ⇒ Hay al menos 13 diferentes tipos de diagramas UML:
  - Diagramas del Modelo de la Estructura Estática del sistema: Captura las cosas y las relaciones estructurales entre las cosas.
    - De Clases, De Componentes, De Implementación, De Objetos, De Paquetes, De composición de la Estructura (N).
  - Diagramas del Modelo Dinámico: Captura como las cosas interactúan para generar la funcionalidad necesaria en el software del sistema:
    - De Actividad, De Interacción, De Casos de Uso, De Estado de Máquina. A su vez los Diagramas de Interacción se subdividen en:
      - De Secuencia, De Comunicación, De Información General sobre la Interacción (N), Diagramas de Temporización.



# *Diagrams*

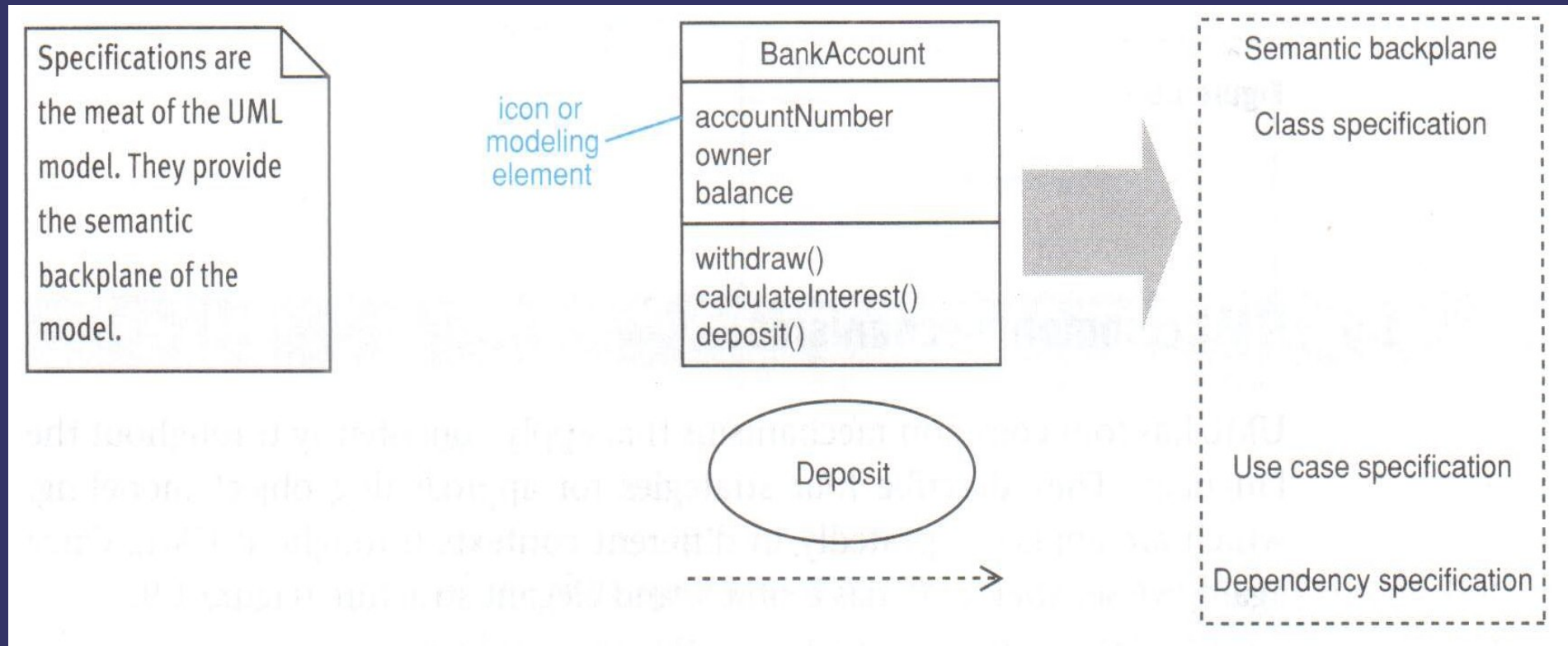


heading syntax: <kind> <name> <parameters>  
N.B. <kind> and <parameters> are optional

# ***Mecanismos Comunes en UML***

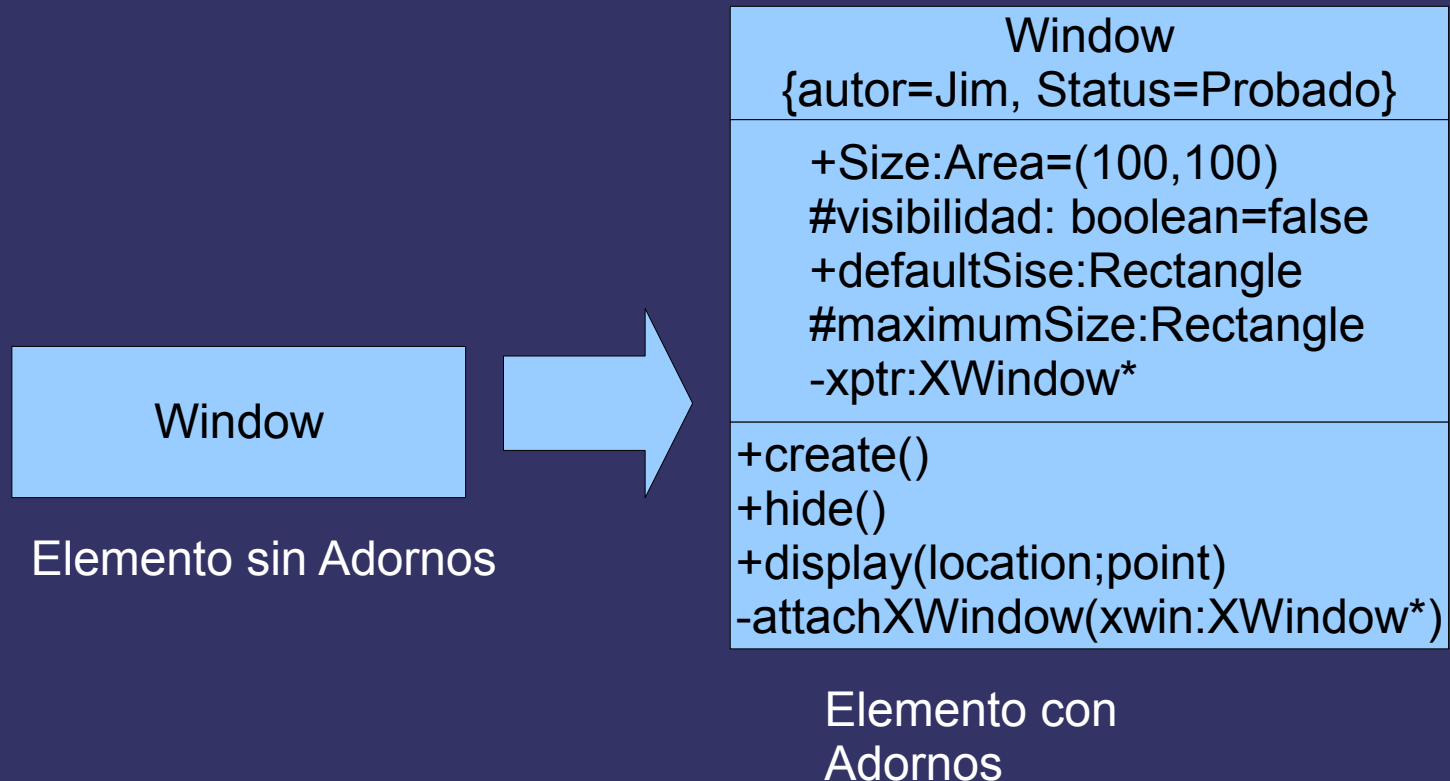
- ⇒ Hay 4 mecanismos comunes que se aplican en UML que describen 4 estrategias para el abordaje del modelado de objetos:
    - **Especificaciones:** tiene al menos 2 dimensiones: una gráfica que permite visualizar diagramas e iconos del modelo y una textual que consiste de especificaciones de varios elementos del modelado
    - **Adornos:** Son los atributos que hacen visibles aspectos de las especificaciones de elementos.
    - **Divisiones Comunes:** La forma de pensar acerca del mundo: Clasificador/Instancia, Interfaz/Implementación.
    - **Mecanismos de Extensibilidad:** 3 mecanismos: restricciones, estereotipos, valores etiquetados, Perfiles UML.
- 
-

# Mecanismos. Especificaciones



- UML permite una gran flexibilidad en la construcción del modelo, En particular los modelos pueden ser:  
elided. Incompleto, inconsistente

# Mecanismos Comunes. Adornos



## ***Divisiones Comunes. Clasificador/Instancia***

Clasificador	Semántica
Actor	Un rol desempeñado por un usuario externo al sistema a quién el sistema enviará un valor
Class	La descripción de un grupo de objetos que comparten alguna función
Componente	Una parte reemplazable o modular de un sistema que encapsula su contenido
Interfaz	Una colección de operaciones que son usadas para especificar un servicio ofrecido por una clase o un componente.
Nodo	Un elemento físico que en tiempo de ejecución representa un recursos
Señal	Un mensaje asíncrono pasado entre objetos
Caso de Uso	Una descripción de una secuencia de acciones que un sistema ejecuta para dar el mejor rendimiento al usuario

La Noción abstracta de un tipo de cosas es el **clasificador** y la cosa concreta específica por ella misma es la **instancia**, Ejm. Una cuenta de banco es un clasificador, pero MI CUENTA DE BANCO es la instancia.

---

---

# ***Mecanismos. Divisiones Comunes.***

## ➞ Interfaz / Implementación.

- Lo importante es separar QUÉ hace una cosa de CÓMO lo hace.
- Una interfaz se define casi como un contrato legal que especifica una garantía de implementación adherida a.
- La interfaz esconde del usuario la complejidad del funcionamiento de una cosa.
- Ej. La interfaz de una impresora láser, está disponible al oprimir el botón de encendido y permite que el usuario configure su impresora de acuerdo a su necesidad y esconde toda la complejidad que conlleva hacer lo mismo en forma manual directamente en el mecanismo de la máquina.

# ***Mecanismos. Mecanismos de Extensibilidad***

- ➔ No es posible diseñar UML para que pueda ser utilizado para todos los posibles fines que a los usuarios se les presenten
- ➔ Por eso se generaron 3 mecanismos de extensibilidad:

Mecanismo	Semántica
Constraints, Restricciones	Éste extiende la semántica de un elemento para permitir agregar nuevas reglas
Estereotipos	Nos permite definir un nuevo elemento de modelado UML basado en otro existente. Nosotros definimos la semántica de nuestro estereotipo. El estereotipo agrega nuevos elementos al metamodelo UML
Tagged Values / Valores por defecto para las propiedades	Proveen una forma de extender las especificaciones de los elementos permitiéndonos agregar uno nuevo, con información adecuada a nuestro propósito en él.

# ***Mecanismos de Extensibilidad.***

## ➔ Constraint / Restricciones.

- Es un texto encerrado entre {} que especifica algunas condiciones o reglas acerca de algún elemento de modelado que debe ser mantenido como verdadero.

## ➔ Estereotipos.

- Representa una variación al modelado de elementos existentes, con los mismos atributos y relaciones pero con otra intención.
- Se puede definir un estereotipo por agregar el nombre del estereotipo entre «..» al nuevo elemento. Puede modelar estereotipos usando el elemento Class con el estereotipo especial predefinido UML «stereotype» que crea un metamodelo de su propio sistema de estereotipos.

## ➔ Valores Etiquetados:

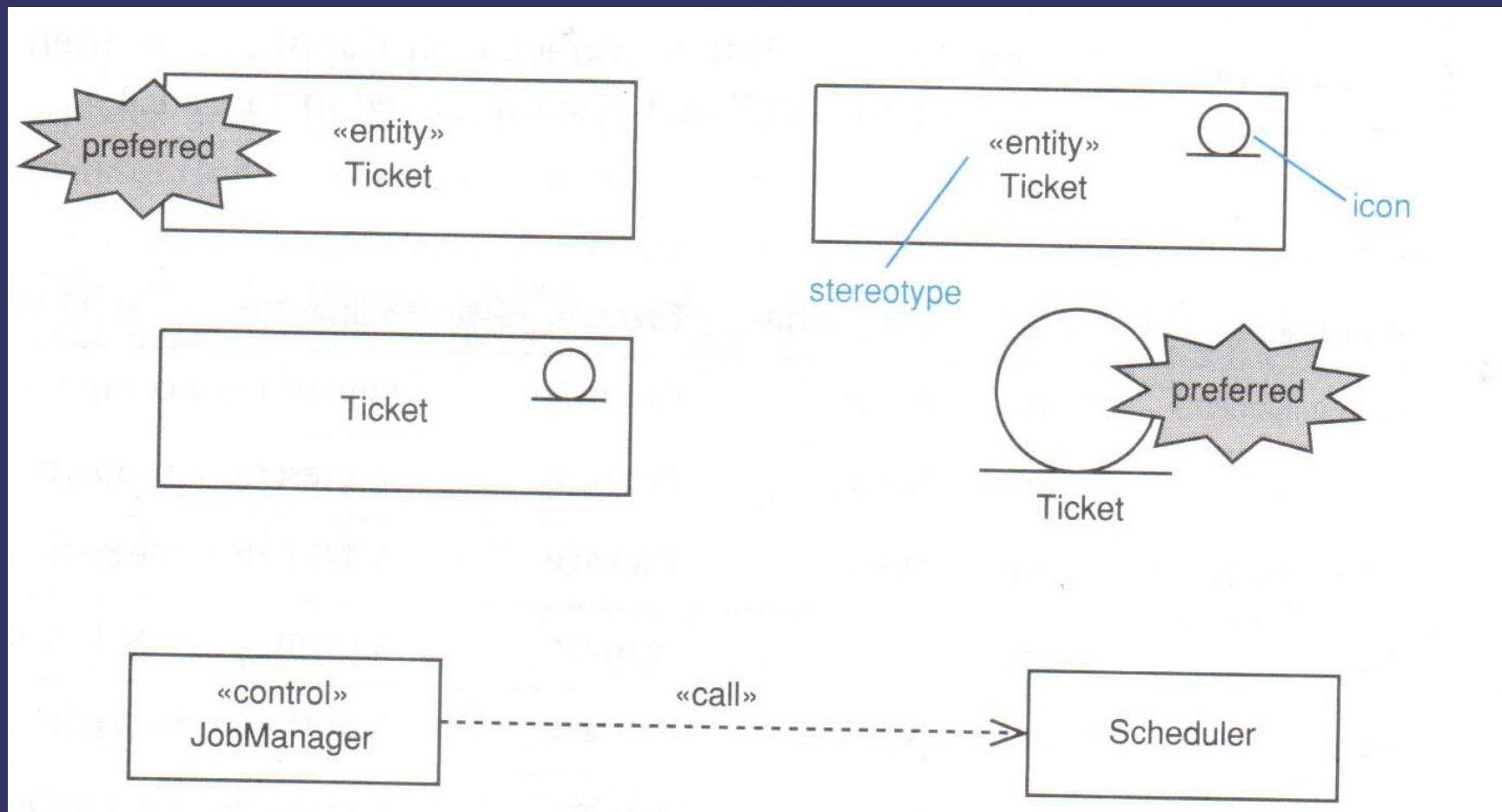
- UML le permite crear sus propios valores etiquetados y asociarlos a un elemento, o crear sus propios atributos y construir estereotipos.

## ➔ UML Profiles / Perfiles UML

- Es una colección de estereotipos, valores etiquetados y restricciones que usted usa para personalizar su UML



# Variantes de Prototipos



# Arquitectura

Define la estructura organizacional del sistema, incluyendo su descomposición en partes, su conectividad, interacción, mecanismos y principios de orientación que informan el diseño de un sistema. Es el concepto de más alto nivel de un sistema en su ambiente.

Clasificador	Función
Vista Lógica	Captura el vocabulario del dominio del problema como un set de clases y objetos
Vista Proceso	Modelo de los subprocesos ejecutables y procesos en un sistema como una clase activa. Es una variación de una vista lógica, con orientación al proceso
Vista Implementación	Los archivos del modelo y componentes que construyen el código físico base de un sistema.
Vista Implementación Física	Modelos de implementación física de artefactos en un set de nodos físicos computacionales como computadoras y periféricos
Vista Casos de Uso	Captura los requerimientos básicos para el sistema como un set de casos de uso, proveen las bases para la construcción de las demás vistas.

# Vistas en la Arquitectura de Software

