

# Arquitectura de Software

Documentación de Arquitecturas  
Parte I

- No es posible describir una Arquitectura de Software con un solo diagrama
- Cada diagrama debe tener una semántica clara para todos los participantes
- Lo que un sistema **debe** hacer es solo una parte del problema, también es importante **cómo** se hace

## □ Las estructuras de un sistema

- **Estáticas** : Definen los elementos del sistema y sus relaciones en tiempo de diseño
  - Ejemplo: módulos, clases, paquetes, tablas.
- **Dinámicas**: Definen los elementos del sistema y sus interacciones en tiempo de ejecución
  - Ejemplo: dataflows y controlflows

- Las propiedades visibles externas
- **Propiedades de Comportamiento:**  
Definen las interacciones funcionales entre un sistema y su ambiente
- **Propiedades de Calidad:** Propiedades no funcionales del sistema tales como desempeño, seguridad y escalabilidad.



“A **Candidate Architecture** for a system is a particular arrangement of static and dynamic structures that has the potential to exhibit the system’s required externally visible behaviors and quality properties” [1]

“An **Architectural Element** is a fundamental piece from which a system can be considered to be constructed” [1]

- Un elemento arquitectural debe:
  - Definir claramente un conjunto de responsabilidades
  - Tener una frontera claramente definida
  - Definir claramente un conjunto de interfaces, las cuales definen los servicios ofrecidos a otros elementos arquitecturales

# Conceptos Básicos

“An **Architectural Pattern** is a description of element and relation types together with a set of constraints on how they may be used”[2]

Un patrón puede ser visto como un conjunto de restricciones en una arquitectura

- Presentan atributos de calidad conocidos
- Algunos patrones representan soluciones conocidas a requerimientos de calidad
- También Conocidos como Estilos Arquitecturales



A **view** is a representation of a coherent set of architectural elements, as written by and read by system stakeholders” [2]

A **structure** is the set of elements itself, as they exist in software or hardware” [2]

# Conceptos Básicos

“**Viewpoints** are an approach to structuring the Architecture Definition process and the Architecture Description, based on the principle of separation of concerns” [1]

“**Perspectives** help structure the Architecture Definition process by separating concerns but focusing on cross-structural quality properties rather than architectural structures” [1]

- “Architects need to think about their software in three ways simultaneously:
  - How it is structured as a set of implementation units
  - How it is structured as a set of elements that have runtime behavior and interactions
  - How it relates to nonsoftware structures in its environment”

“Documenting software Architectures” pag 18

- Las vistas arquitecturales están clasificadas en tres tipos (viewtypes):
  - Module
  - Component and Connector
  - Allocation

## ❑ Module

- ❑ Las unidades de descomposición son módulos
- ❑ Unidades de código que implementan un conjunto de responsabilidades
  - ❑ Clase, Colección de clases, Capas
- ❑ La relación “sub modulo” está presente

- Component and Connector
  - Modela los aspectos dinámicos del sistema en ejecución
  - Las unidades son procesos
  - La relación entre los elementos representa la comunicación entre los componentes y los conectores

## ❑ Allocation

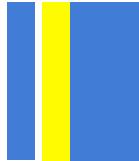
- ❑ Modela las estrategias de asignación entre elementos
- ❑ Los elementos pueden ser software, hardware, o canales de comunicación
- ❑ La relación “asignado en” es utilizada entre los elementos

Un **Estilo Arquitectural** expresa una estructura fundamental o un esquema organizacional de un sistema de software [1]

Un **Estilo** define una familia de arquitecturas que satisfacen algunas restricciones [2]

Un estilo define:

- Una familia de sistemas cuya estructura es la misma.
- Un vocabulario para los componentes
- Un vocabulario para los estilos
- Un patrón de arquitectura



# Conceptos Básicos

- Para cada Estilo Arquitectural es importante tener en cuenta
  - Propósito del estilo
  - Elementos que se pueden utilizar y sus propiedades
  - Relaciones entre los elementos
  - Notaciones apropiadas para dicho estilo (por ejemplo Diagramas UML, etc.)



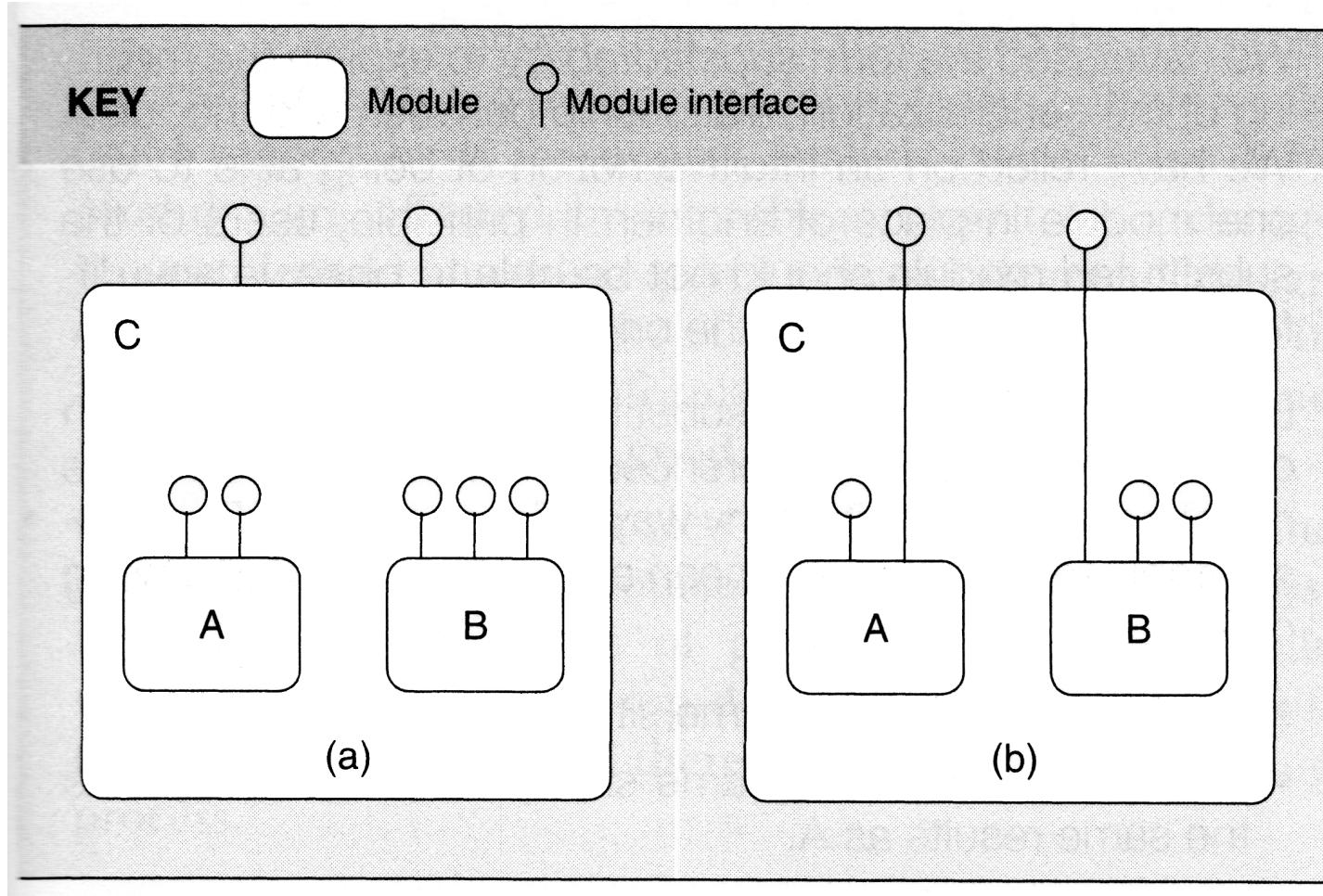
# Conceptos Básicos

## Module Viewtype

Elementos	Módulo
Relaciones	Is-part-of Is-a Depends-on
Propiedades	Nombre Responsabilidades del módulo
Topología	No tiene restricciones
Usos	<ul style="list-style-type: none"><li>•Provee una maqueta del código fuente</li><li>•Trazabilidad de requerimientos</li><li>•Análisis de Impacto</li><li>•Permite explicar la funcionalidad del sistema a los stakeholders</li></ul>
Notaciones	<ul style="list-style-type: none"><li>•Informales</li><li>•UML (Clases, paquetes)</li></ul>

# Conceptos Básicos

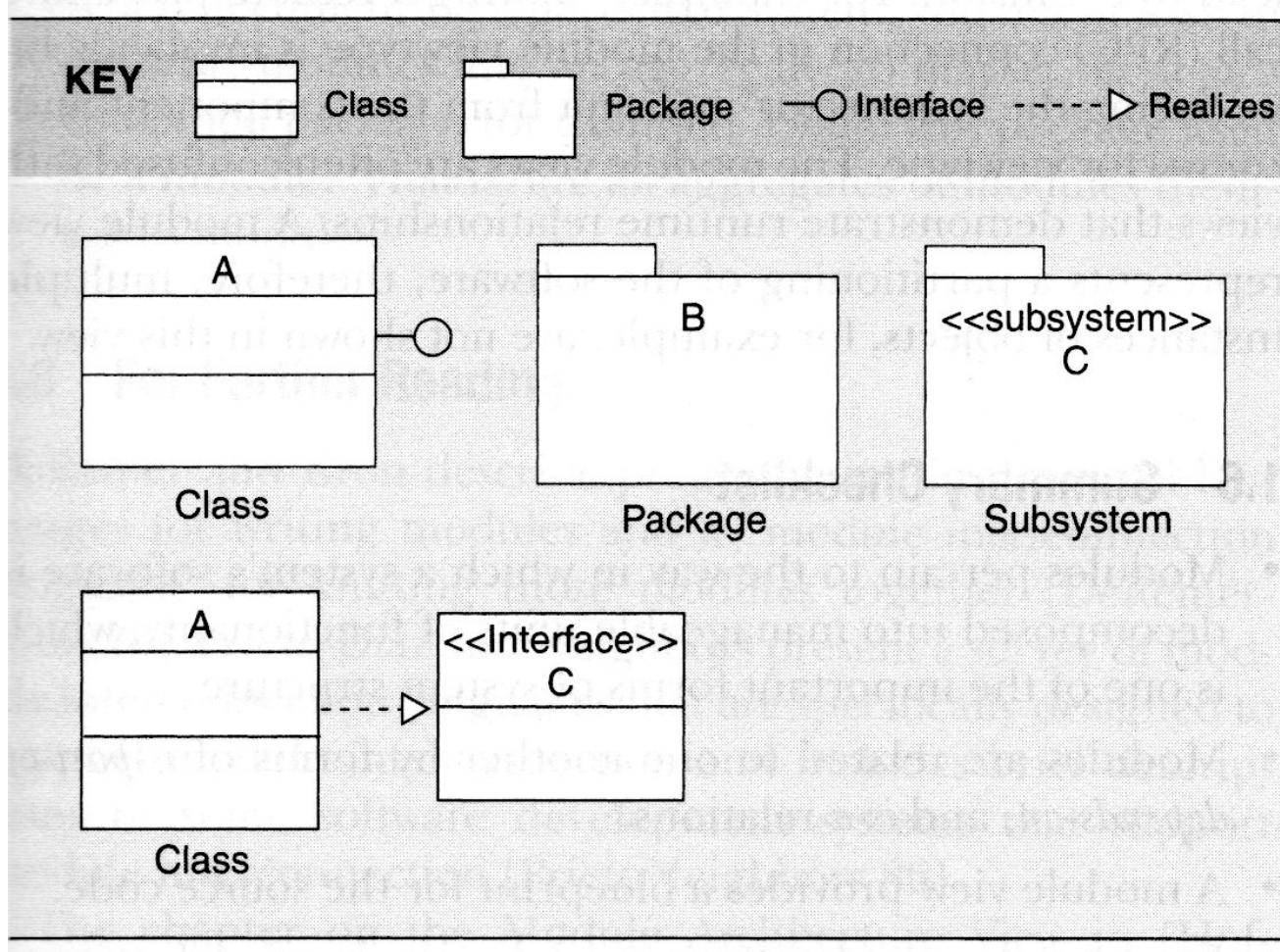
## Ejemplos de notaciones para el tipo de vista *Module*



Tomado de “Documenting Software Architectures” pag 45

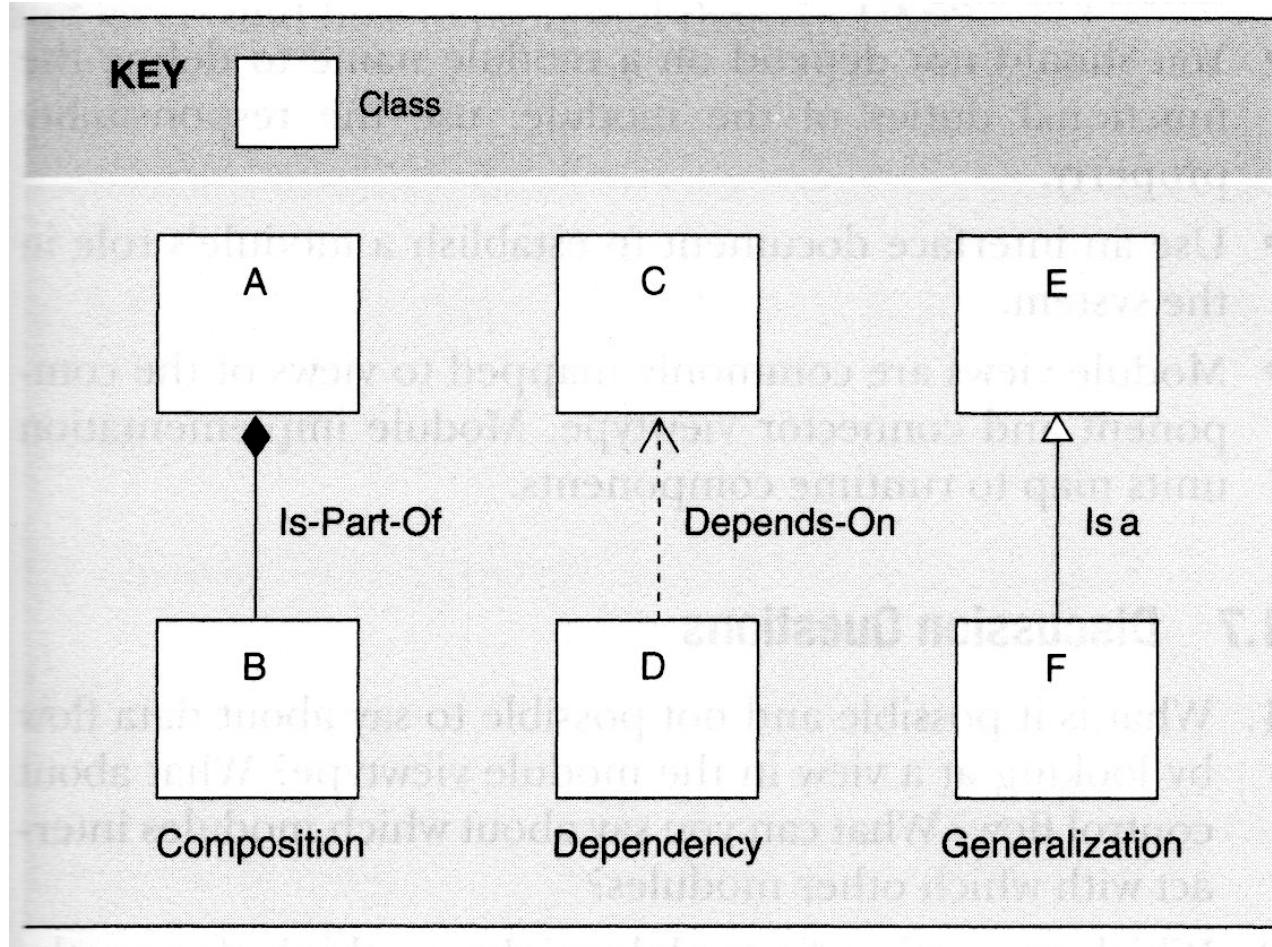
# Conceptos Básicos

Ejemplos de notaciones para el tipo de vista *Module*



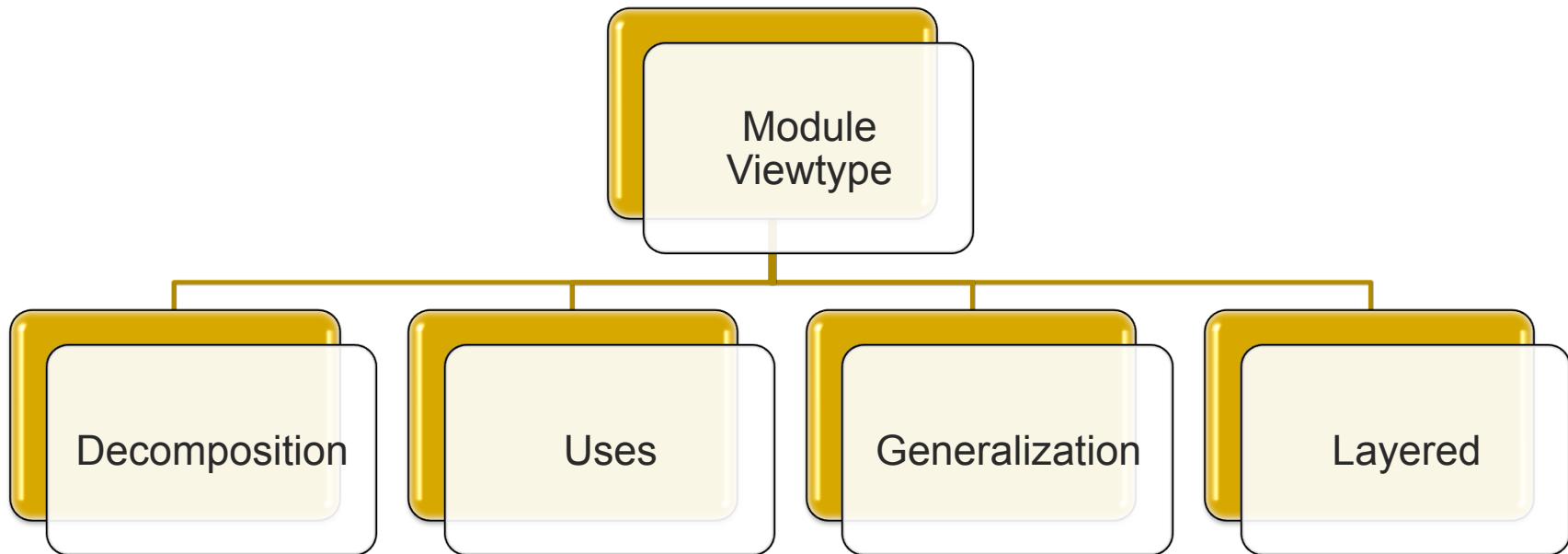
# Conceptos Básicos

Ejemplos de notaciones para el tipo de vista *Module*



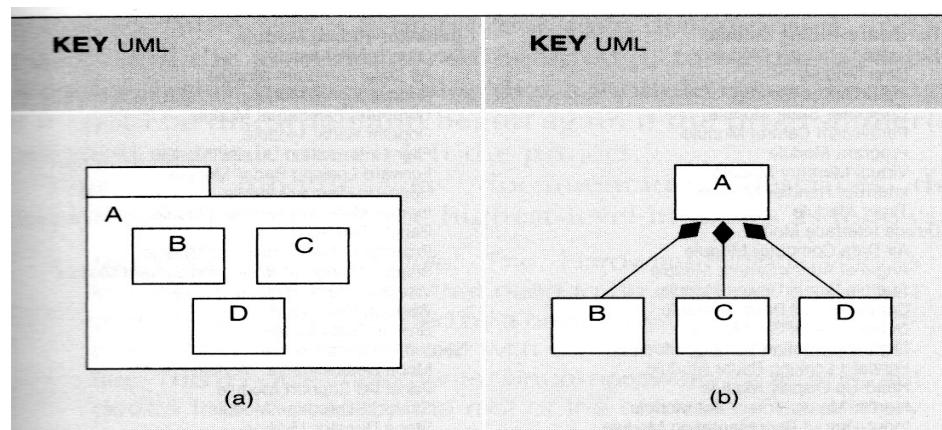
Tomado de “Documenting Software Architectures” pag 49

# Conceptos Básicos



# Conceptos Básicos

Decomposition Style	
Elementos	Módulos
Relaciones	is-part-of
Propiedades	
Propiedades de las relaciones	Visibilidad
Topología	No se permiten ciclos Un módulo solo es parte de otro módulo
Notación	UML



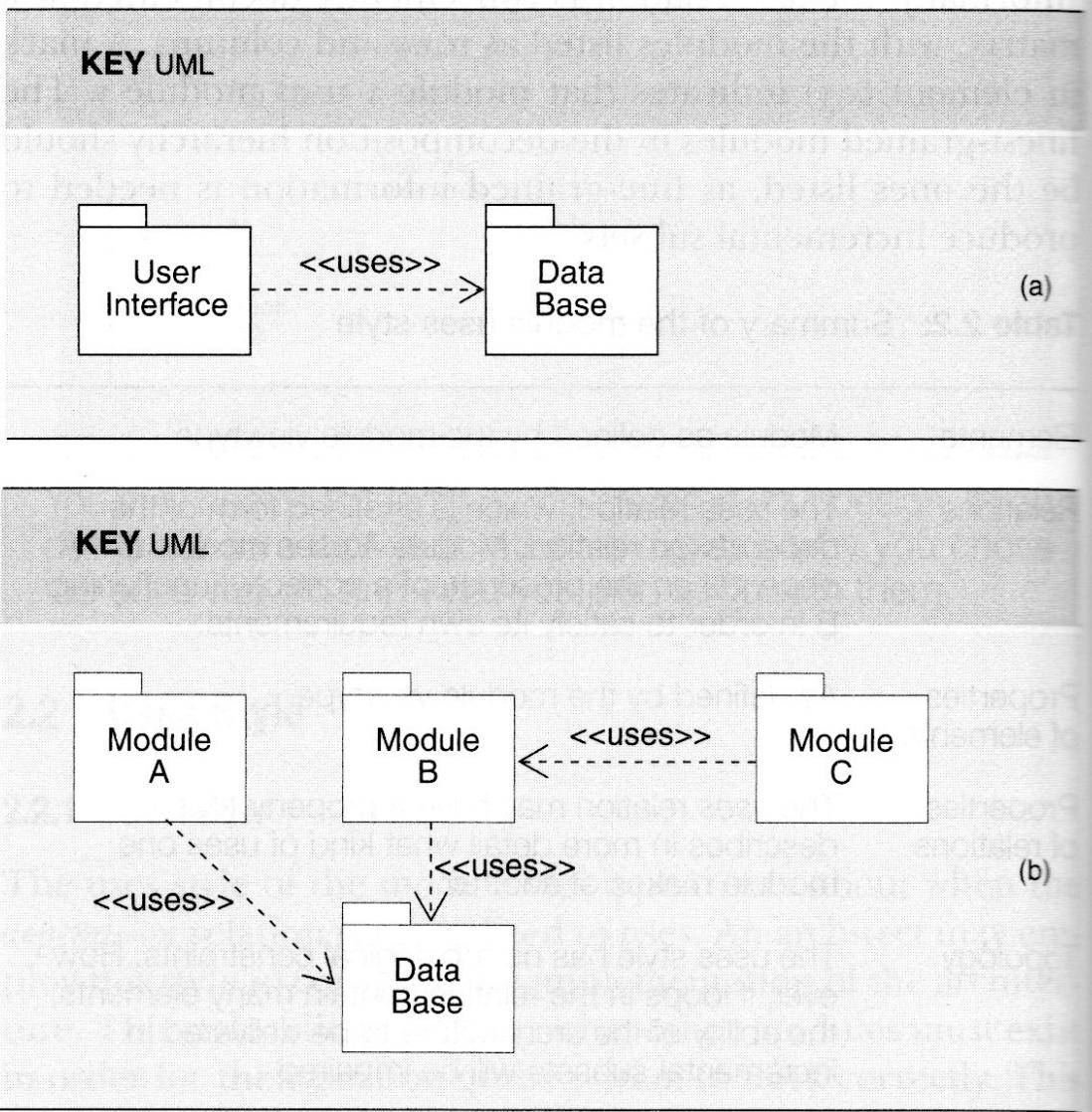
Tomado de "Documenting Software Architectures" pag 57



# Conceptos Básicos

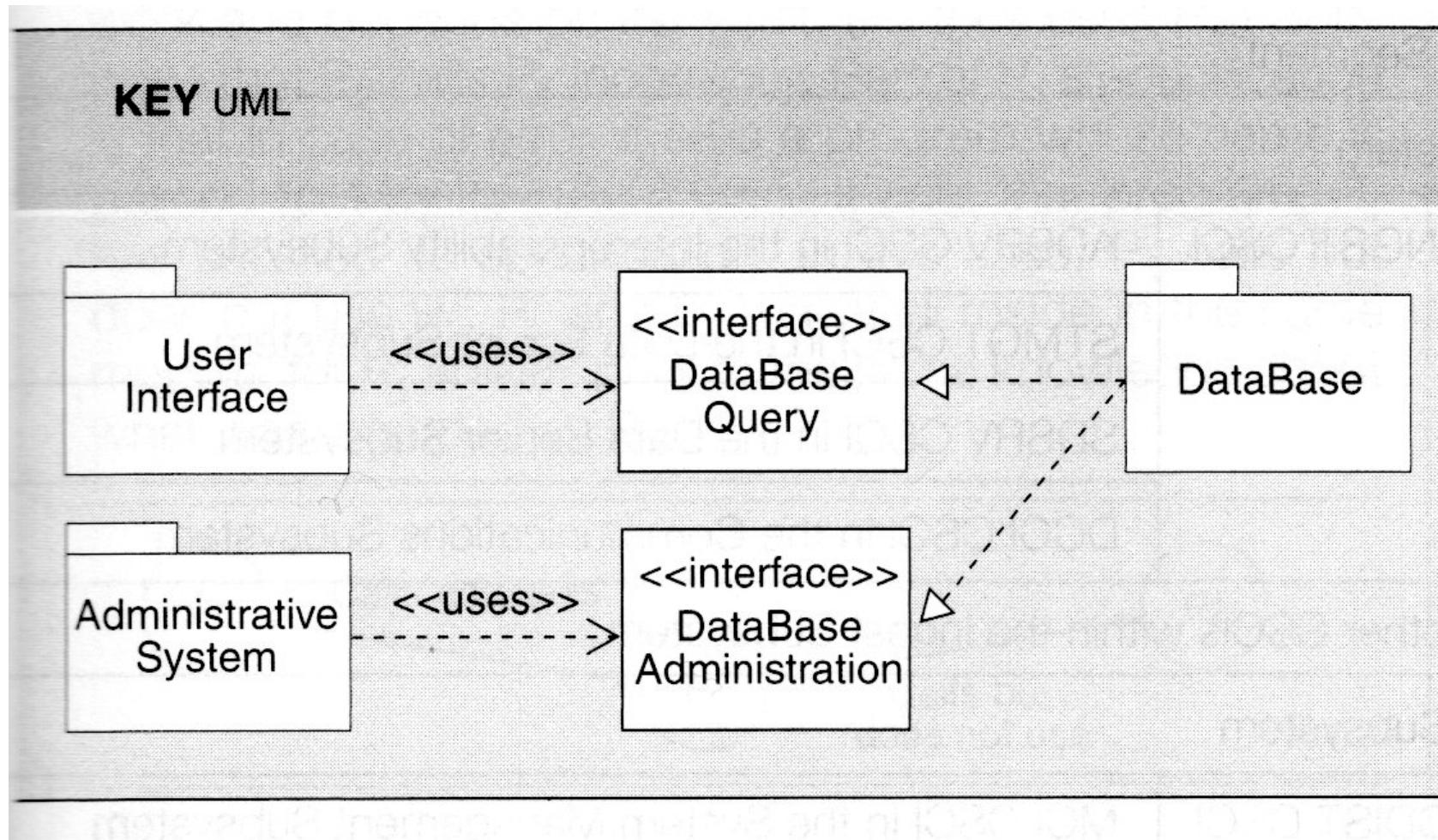
Uses Style	
Elementos	Módulos
Relaciones	depends-on
Propiedades	
Propiedades de las relaciones	
Topología	
Notación	UML

# Conceptos Básicos



Tomado de “Documenting Software Architectures” pag 66

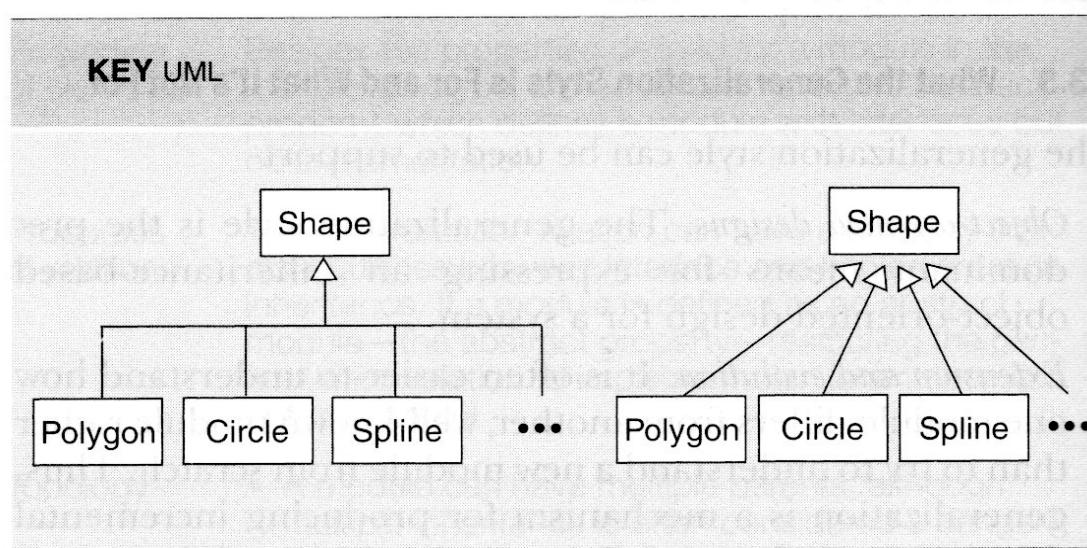
# Conceptos Básicos



Tomado de "Documenting Software Architectures" pag 67

# Conceptos Básicos

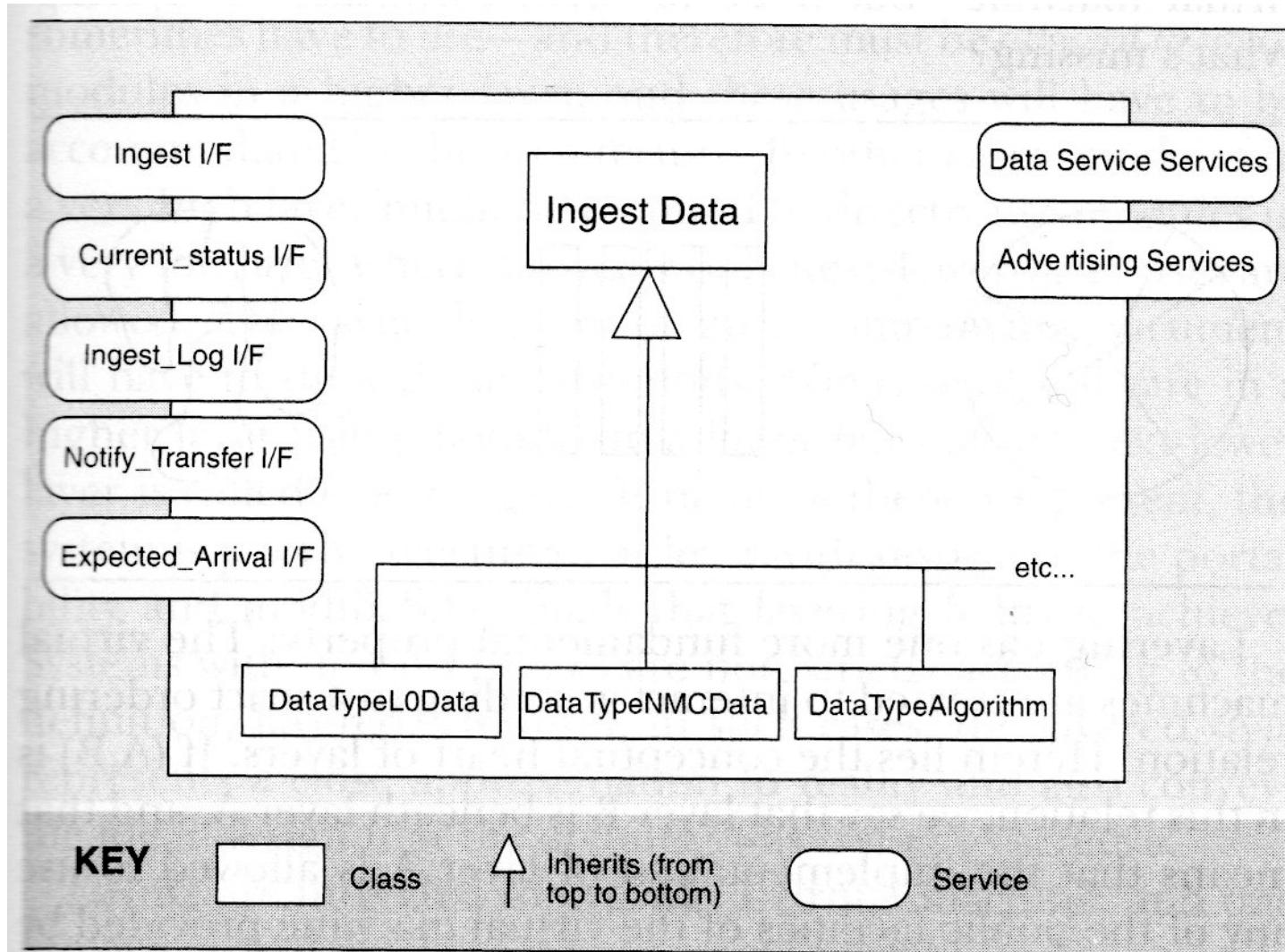
Generalization Style	
Elementos	Módulos
Relaciones	Generalization
Propiedades	
Propiedades de las relaciones	Diferenciar entre interfaces e implementación
Topología	Se permite herencia múltiple
Notación	UML



Tomado de "Documenting Software Architectures" pag 74



# Conceptos Básicos



Tomado de "Documenting Software Architectures" pag 77



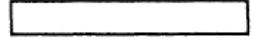
# Conceptos Básicos

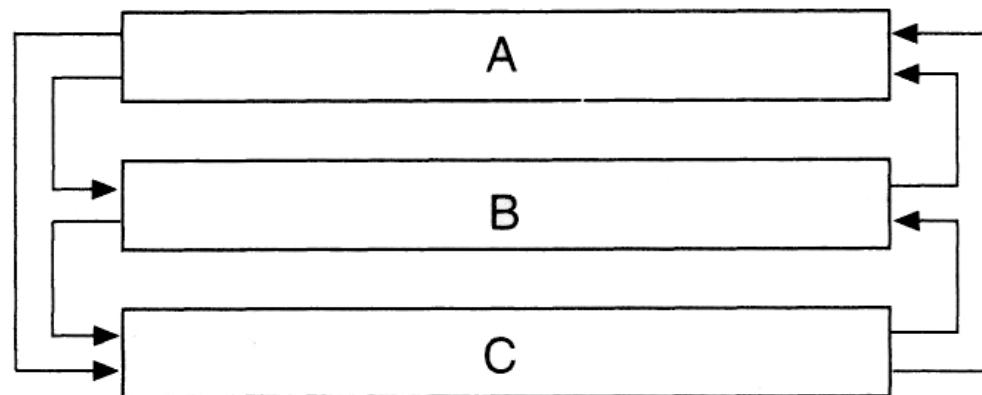
## Layered Style

Elementos	Capas
Relaciones	Autorizado para usar
Propiedades	Nombre de la capa
Propiedades de las relaciones	Diferenciar entre interfaces e implementación
Topología	No se permite el intercambio de posiciones en la jerarquía
Notación	Informales Segmentadas Anillos UML (mediante paquetes)

Ejemplo de notación informal para representar un estilo por capas

---

**KEY** (informal notation)     Layer     $x \rightarrow y$      $x$  is allowed to use  $y$

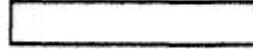


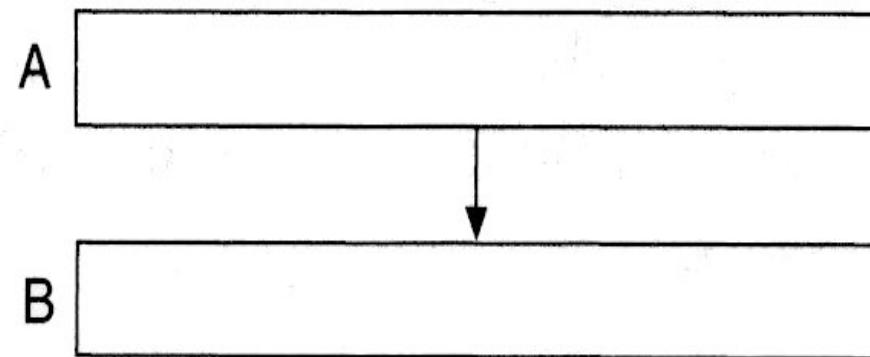
---

Nota algo raro en este ejemplo?

# Conceptos Básicos

Ejemplo de notación informal para representar un estilo por capas

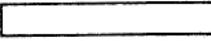
KEY (informal notation)     Layer     $x \rightarrow y$      $x$  is allowed to use  $y$

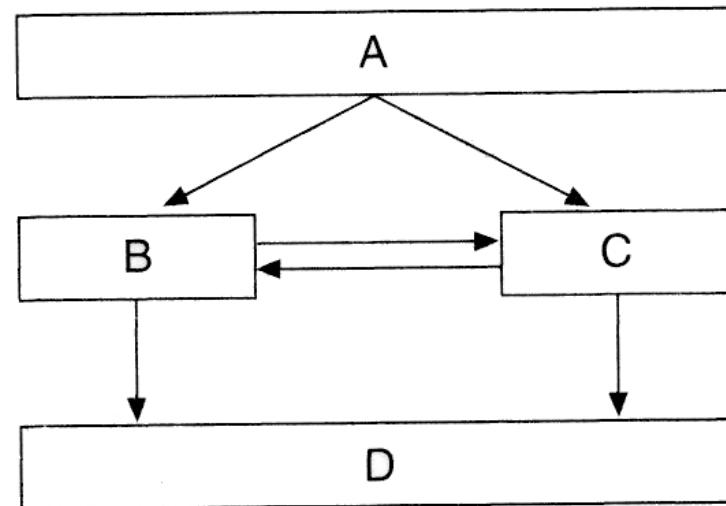


Tomado de "Documenting Software Architectures" pag 83

# Conceptos Básicos

Ejemplo de notación informal para representar un estilo por capas

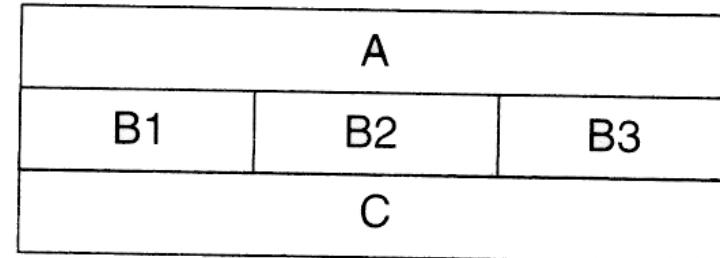
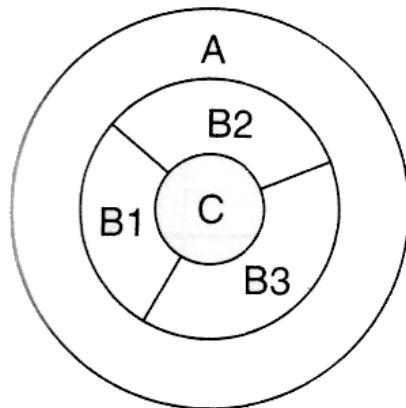
**KEY** (informal notation)     Layer     $x \rightarrow y$      $x$  is allowed to use  $y$



Tomado de “Documenting Software Architectures” pag 84

# Conceptos Básicos

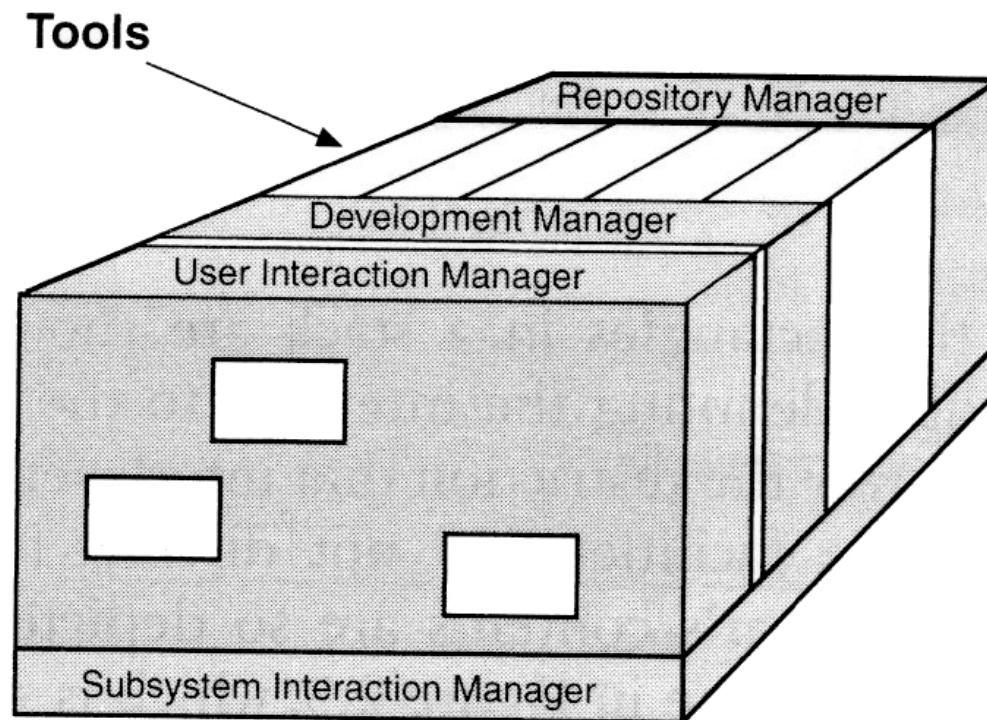
Ejemplo de notación informal para representar un estilo por capas



Tomado de "Documenting Software Architectures" pag 85

# Conceptos Básicos

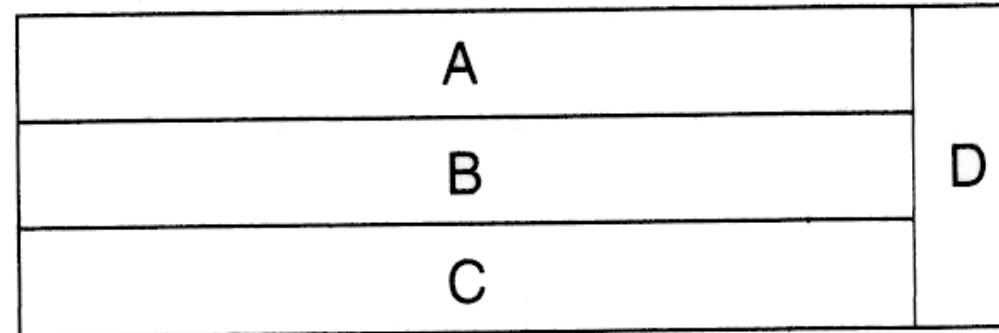
Ejemplo de notación informal para representar un estilo por capas



Tomado de “Documenting Software Architectures” pag 85

# Conceptos Básicos

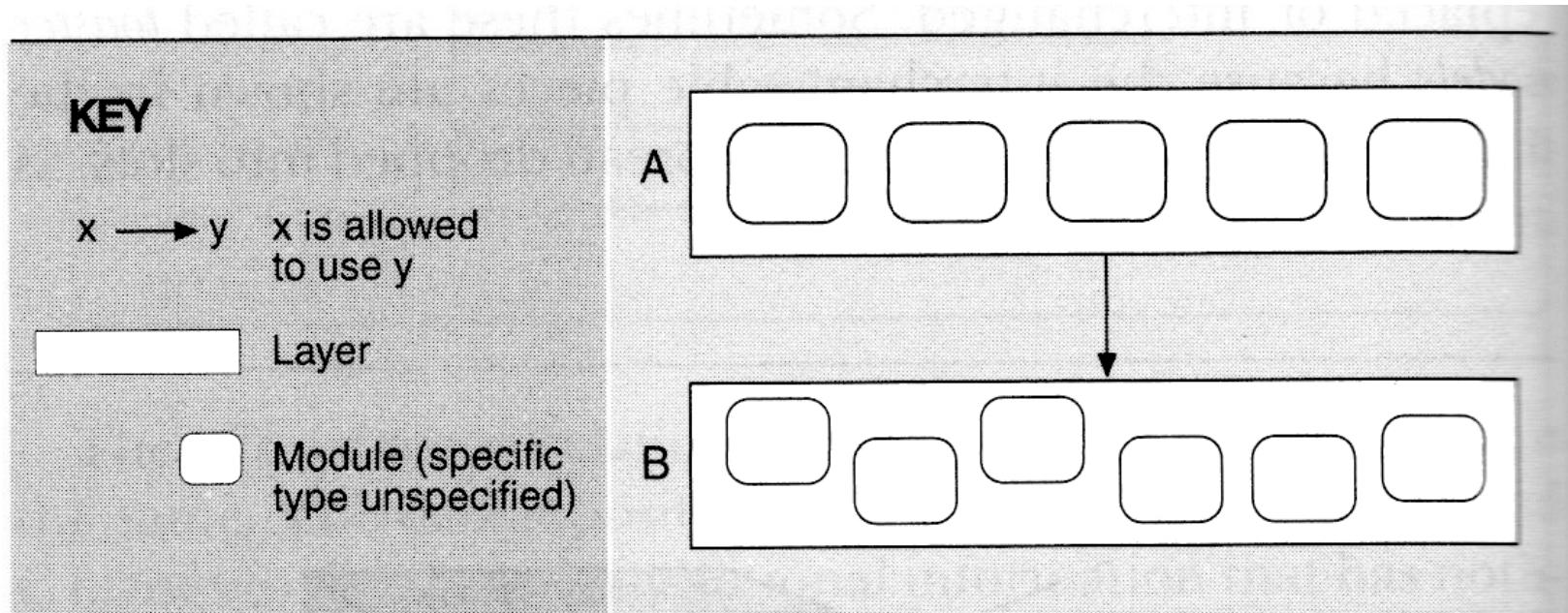
Ejemplo de notación informal para representar un estilo por capas



Tomado de "Documenting Software Architectures" pag 86

# Conceptos Básicos

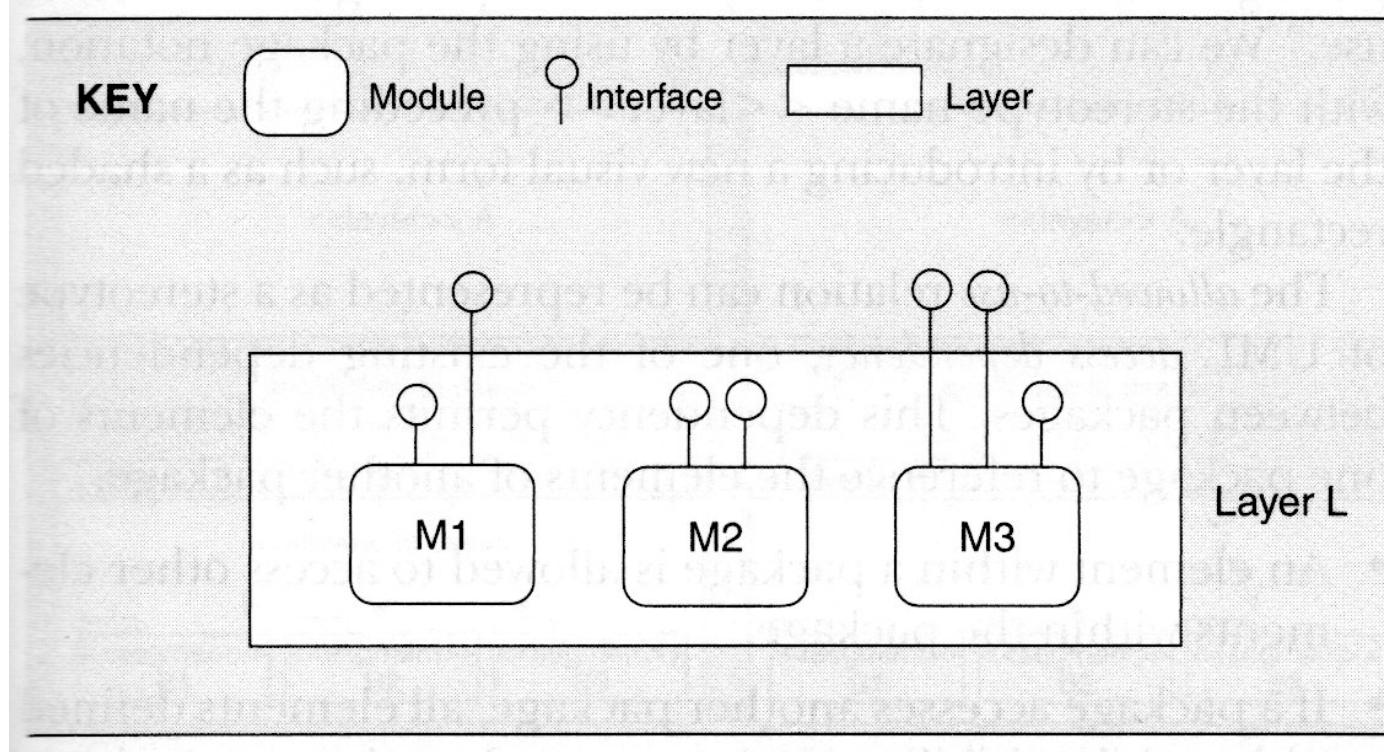
Ejemplo de notación informal para representar un estilo por capas



Tomado de "Documenting Software Architectures" pag 86

# Conceptos Básicos

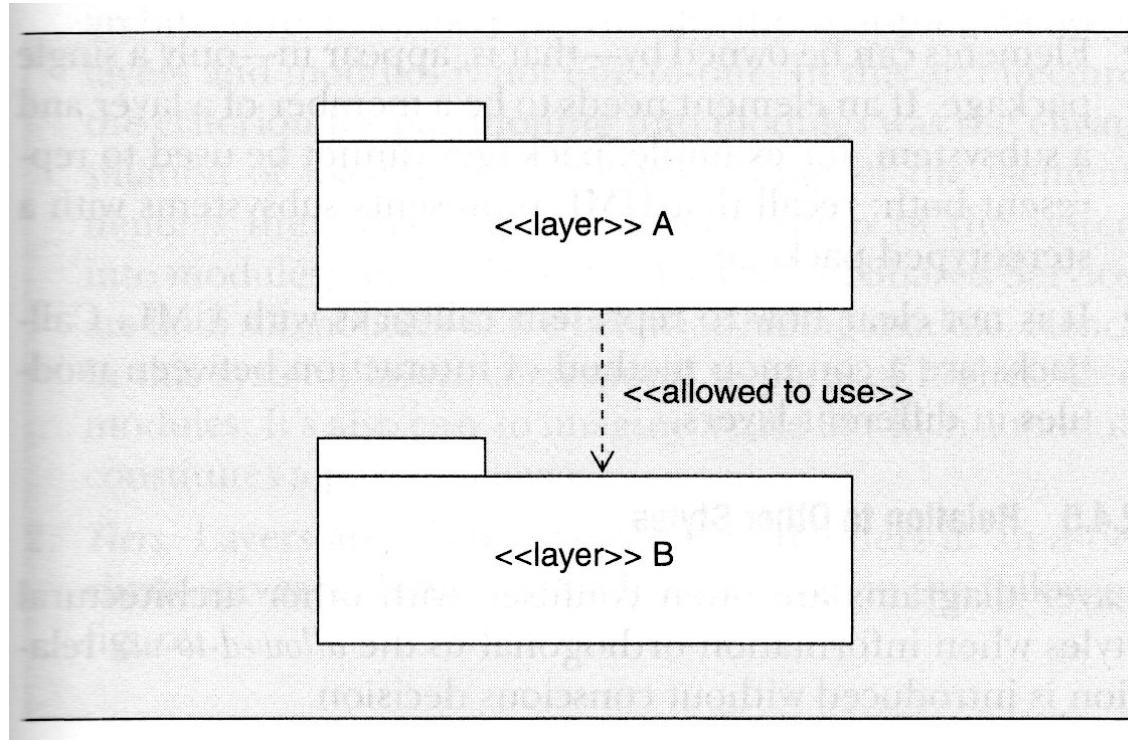
Ejemplo de notación informal para representar un estilo por capas



Tomado de “Documenting Software Architectures” pag 87

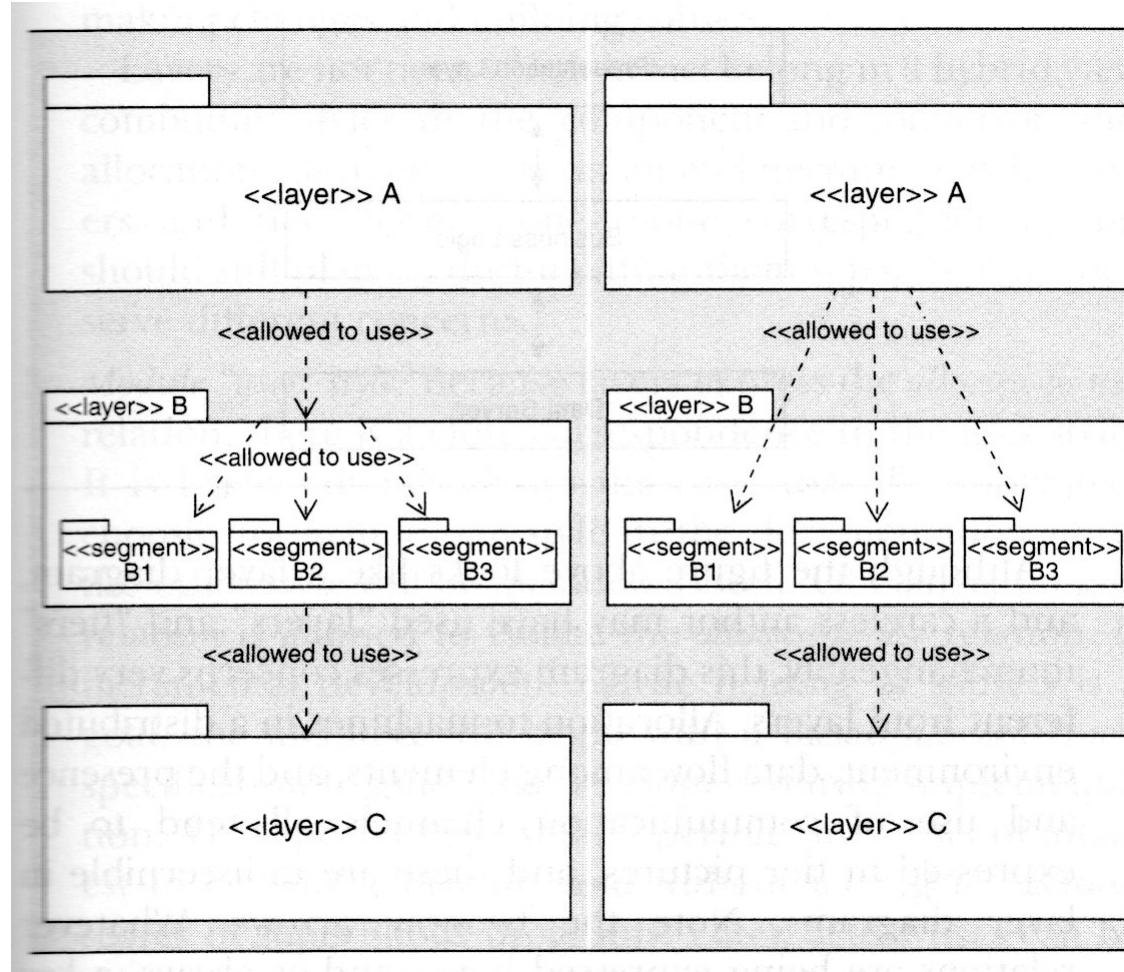
# Conceptos Básicos

Ejemplo de notación UML para representar un estilo por capas



Tomado de "Documenting Software Architectures" pag 87

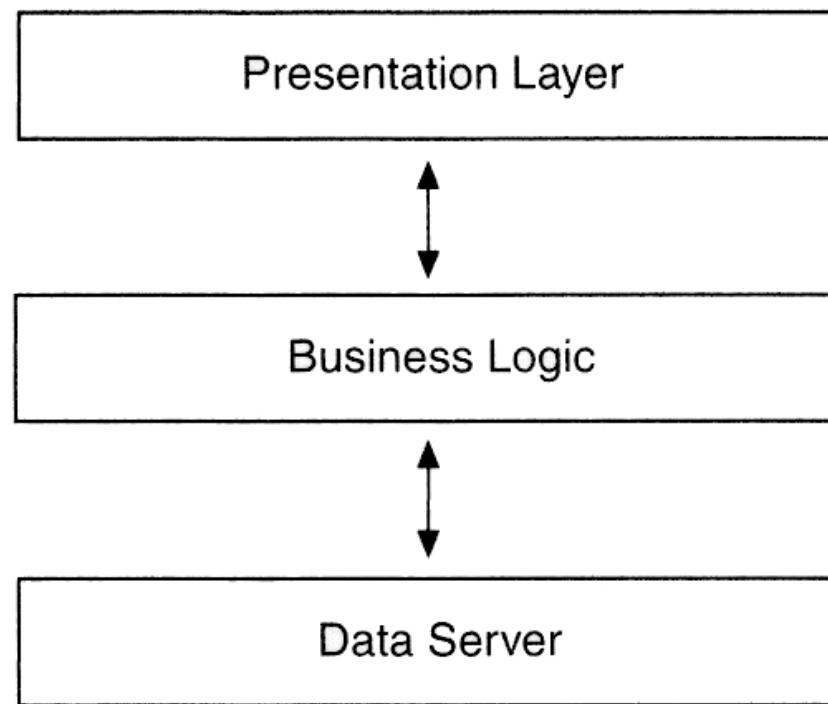
Ejemplo de notación UML para representar un estilo por capas



Tomado de “Documenting Software Architectures” pag 89

# Conceptos Básicos

Qué opina sobre este ejemplo de un estilo arquitectural por capas?

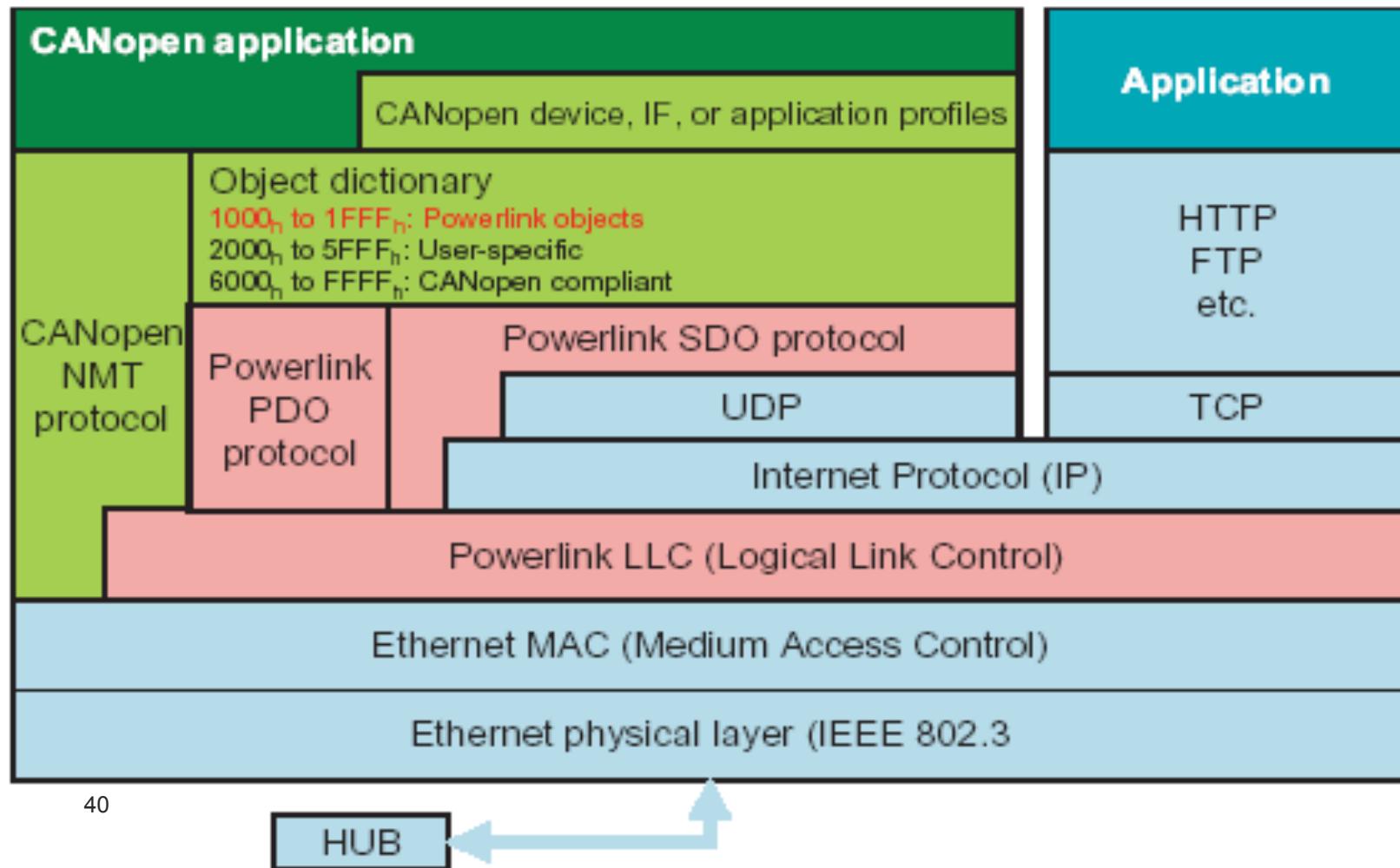


Tomado de “Documenting Software Architectures” pag 90



# Conceptos Básicos

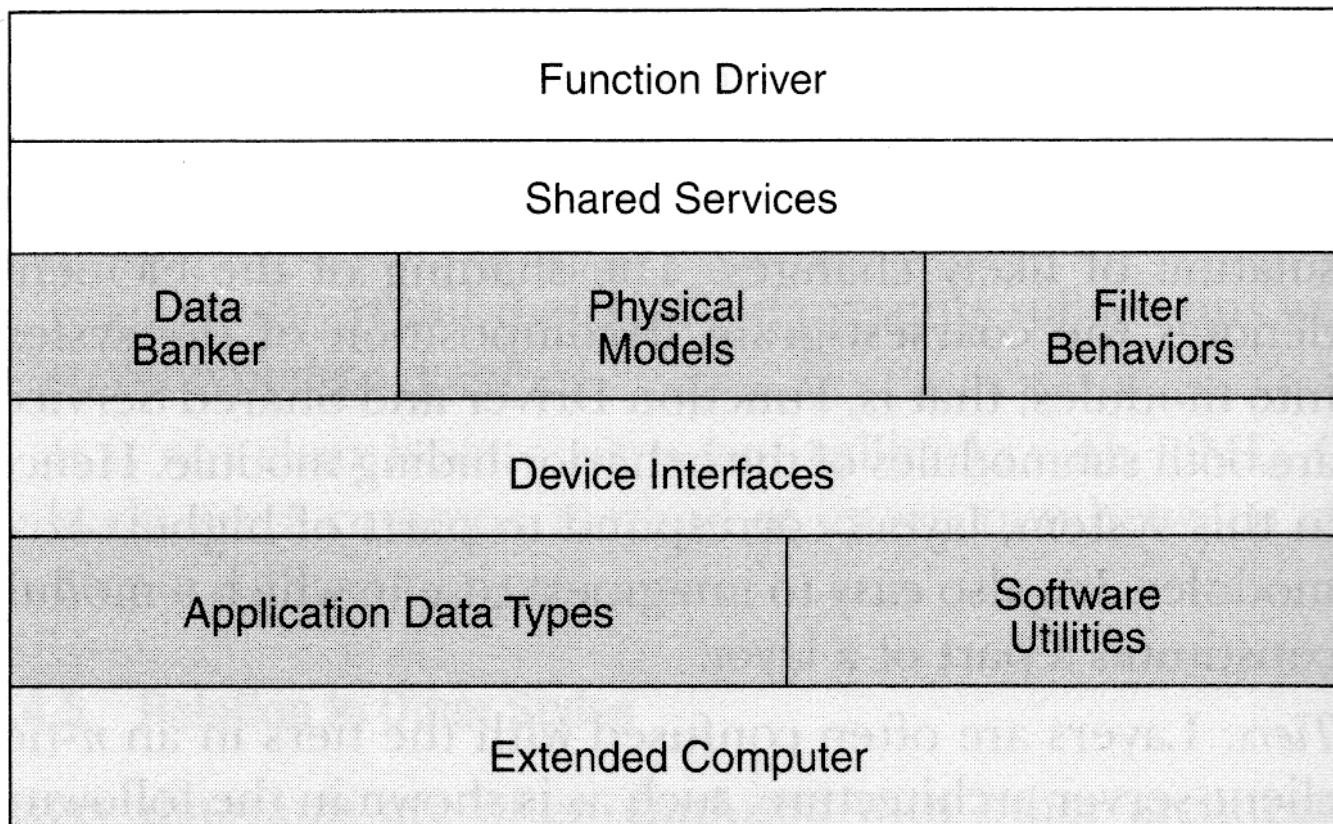
Ejemplo de notación informal para representar un estilo por capas



# Conceptos Básicos

Ejemplo de notación informal para representar un estilo por capas

**KEY**     Behavior-hiding module     Software decision-hiding module     Hardware-hiding module



Tomado de "Documenting Software Architectures" pag 90



## Bibliografía

- [1] Rozanski N, Woods E. “Software Systems Architecture” Addison-Wesley. 2005
- [2] Clements, Paul et al. “Documenting Software Architectures: views and Beyond”. Addison-Wesley. 2002
- [3] Len Bass et Al. “Software Architecture in practice” Second Edition, Addison Wesley, 2007