



# Diagramas en la fase de diseño

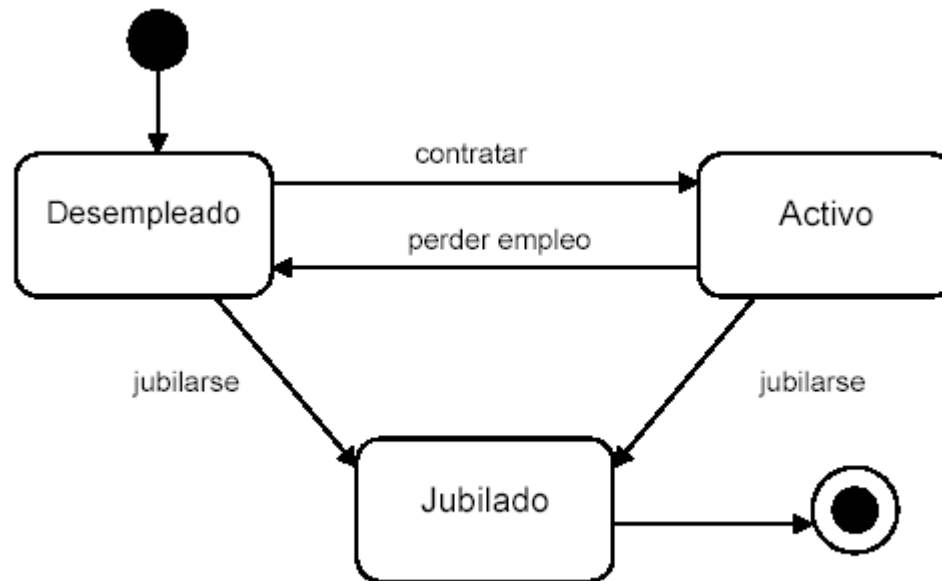
# Diagramas UML

- **Clases** (relaciones estáticas, operaciones, navegabilidad)
- **Objetos** (representa una instancia específica de un diagrama de clases en un momento determinado en el tiempo)
- **Paquetes** (visión de alto nivel mostrando dependencias)
  - mecanismo para agrupación de elementos
  - la dependencia significa que cambios en uno afectan al otro
- **Subsistemas** (visión de paquetes, comportamiento de clases)
  - agrupación lógica de elementos de diseño, con interface de servicios que brinda (implementados por las clases)
- **De Interacción** (Los diagramas UML de secuencia y de colaboración se utilizan para modelar los aspectos dinámicos de un sistema)
  - **Secuencia** (deja en claro la evolución temporal)
  - **Comunicación/Colaboración** (muestra más claramente las interacciones, pero el orden está dado por números)
- **Componentes** (dependencias entre componentes)
- **Despliegue** (Deployment del software en el hardware)
- **Estados** (Modela aspectos dinámicos del sistema y sólo se construye para *objetos reactivos*)

# Diagrama de estados

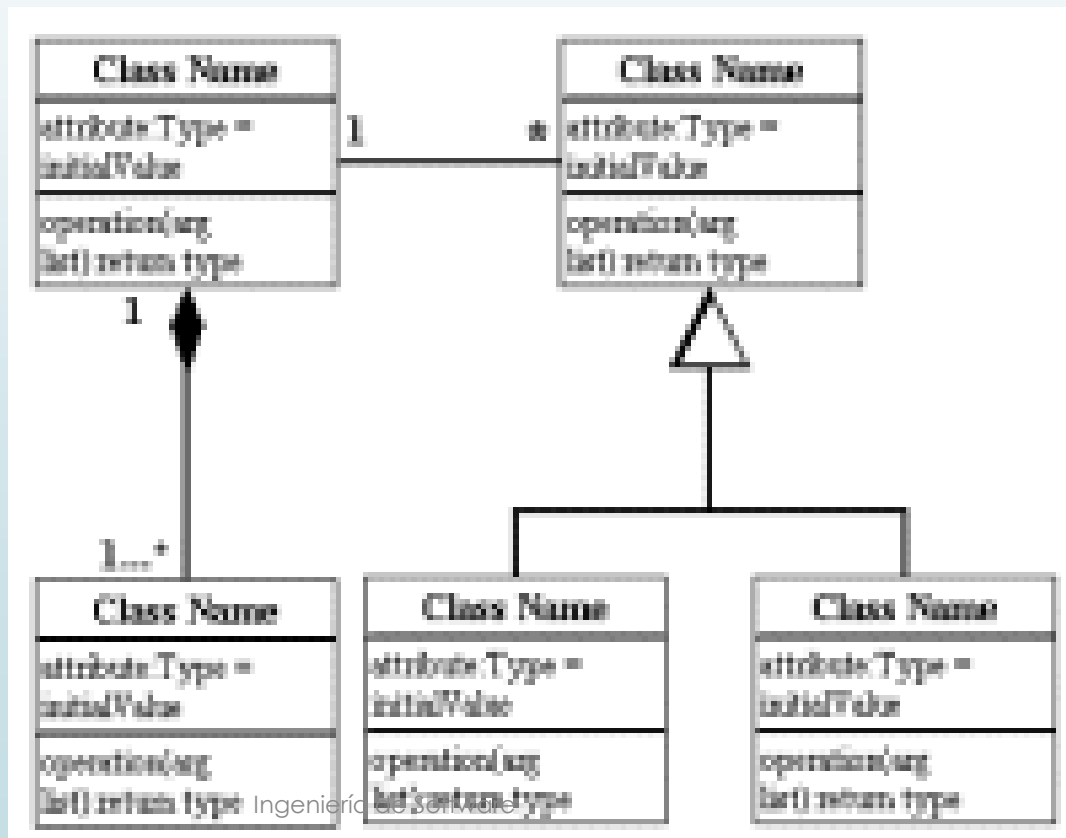
Modela aspectos dinámicos del sistema y sólo se construye para *objetos reactivos*.

Modela el comportamiento individual de cada objeto durante su ciclo de vida.



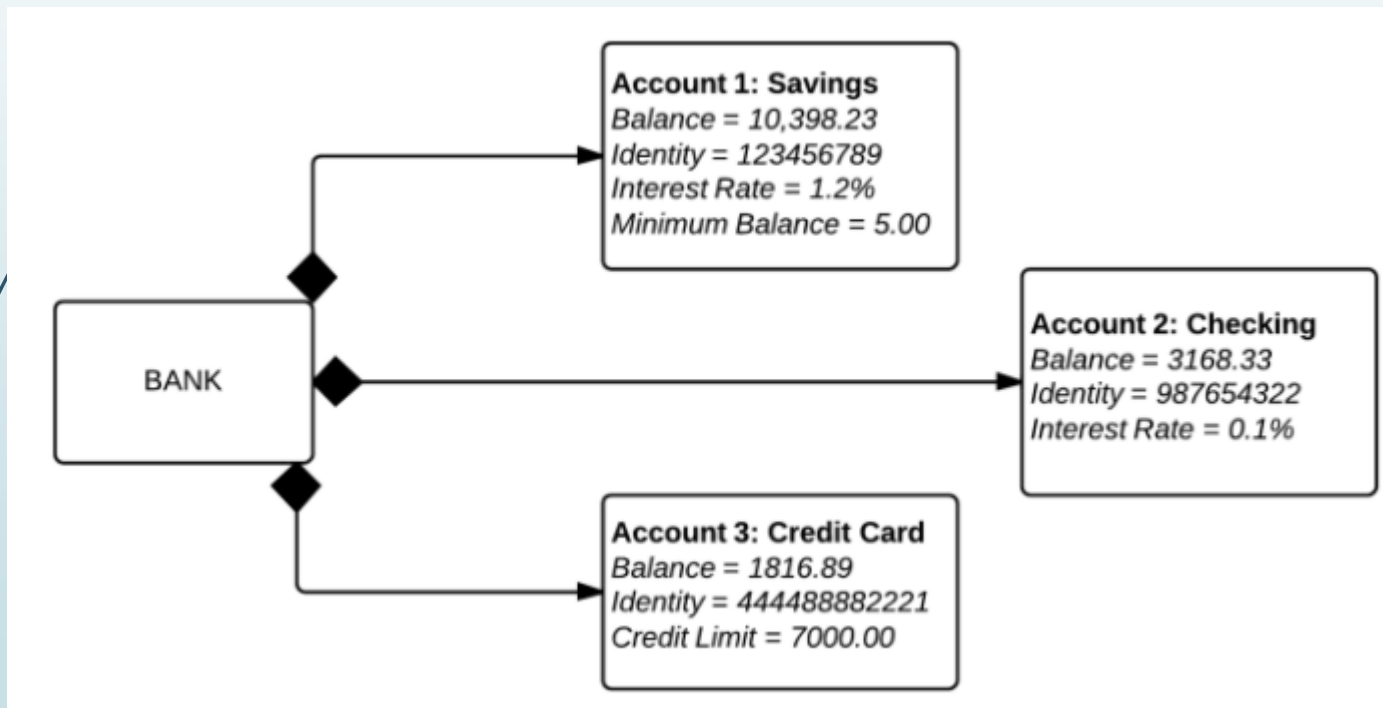
# Diagrama de Clases

- Muestra las clases en el sistema y como se relacionan
- Información sobre atributos y tipos correspondientes, operaciones con parámetros y tipos correspondientes, multiplicidad de las asociaciones, agregación, composición, generalización.



# Diagrama de Objetos

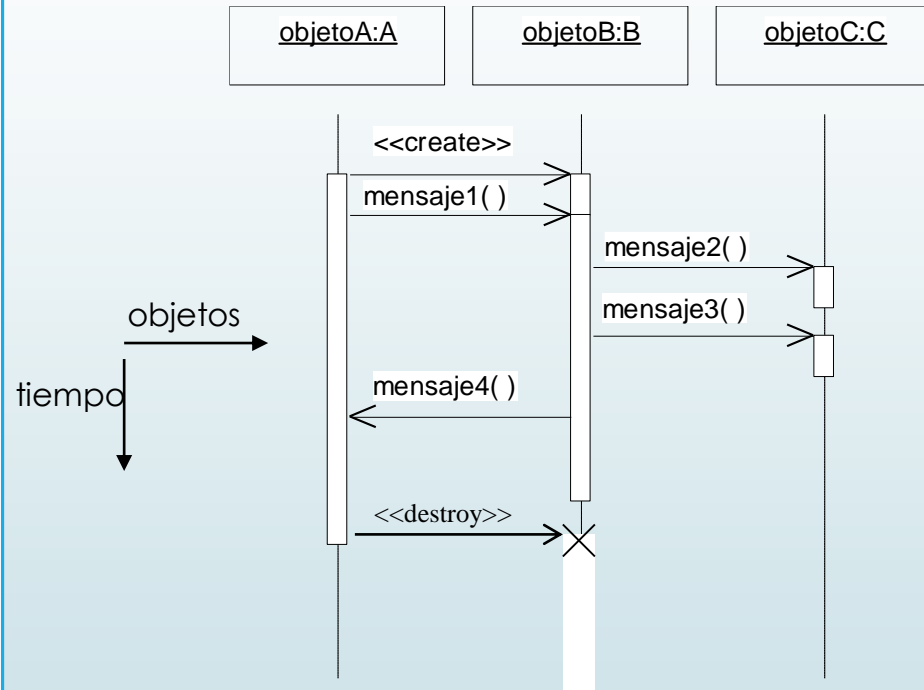
- Un diagrama de objetos se enfoca en los atributos de un conjunto de objetos y cómo esos objetos se relacionan entre sí.
- Por ejemplo, en el siguiente diagrama de objetos, las tres cuentas bancarias están ligadas al banco mismo. Los títulos de clase muestran el tipo de cuentas (ahorros, corriente y tarjeta de crédito) que un cliente dado podría tener con este banco en particular. Los atributos de clase son diferentes para cada tipo de cuenta. Por ejemplo, el objeto de tarjeta de crédito tiene un límite de crédito, mientras que las cuentas de ahorros y corriente tienen tasas de interés



# Diagrama de Interacción: Secuencia

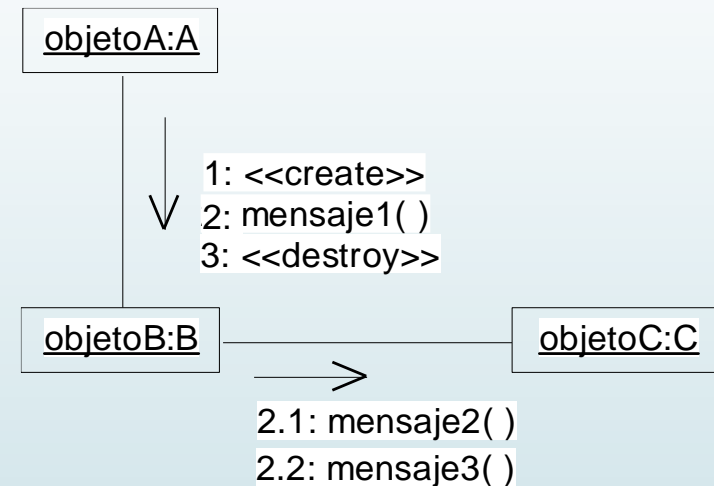
## De Interacción: Secuencia

- Los diagramas de secuencia muestran como los objetos se intercambian mensajes a lo largo del tiempo
- El orden de los mensajes se muestra en relación al tiempo
- Cada objeto tiene una línea de vida sobre la cual se muestran los bloques de activación (tiempo que el objeto necesita para completar una tarea)
- Se pueden modelar mensajes sincrónicos y asincrónicos, loops, entre otros.



# Diagrama de Interacción: Colaboración/Comunicación

- Consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Destaca la relación estructural entre los objetos que interactúan
- El orden de los mensajes se muestra numerado
- Los mensajes se pueden anidar mostrando la anidación con la numeración, se pueden modelar loops.
- Las asociaciones son las existentes entre los objetos en el diagrama de clases.



# Diagrama de Paquetes

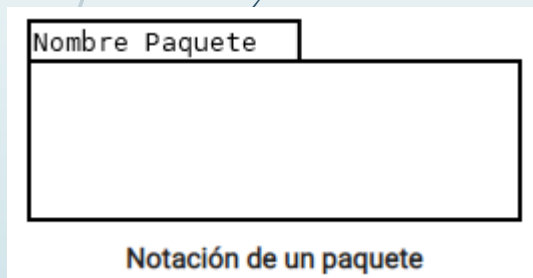
Representa de forma estática los componentes del sistema de información que está siendo modelado.

En concreto puede ser un conjunto de clases, casos de uso, componentes u otros paquetes.

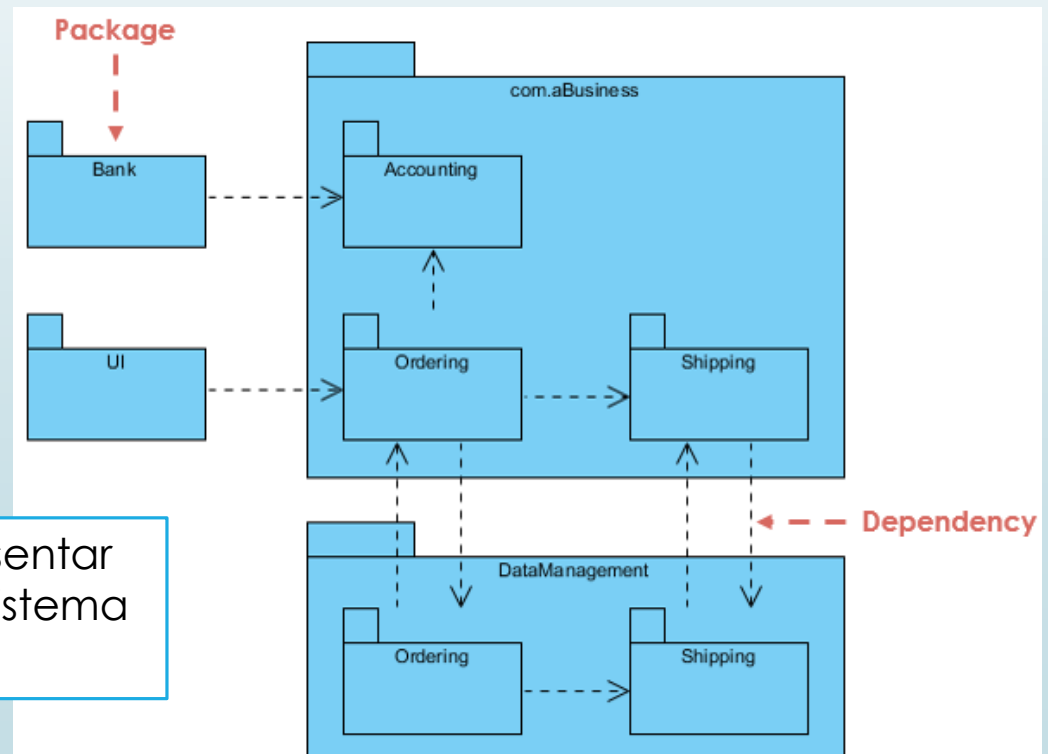
Cada paquete debe tener una función diferenciada del resto de elementos.

Es importante identificar con nombres representativos a los distintos paquetes.

Se representa con un símbolo simulando una carpeta, con el nombre en la parte superior:



Los paquetes pueden representar los diferentes niveles de un sistema para revelar la arquitectura





# Recomendaciones

Se recomienda agrupar en paquetes los elementos que:

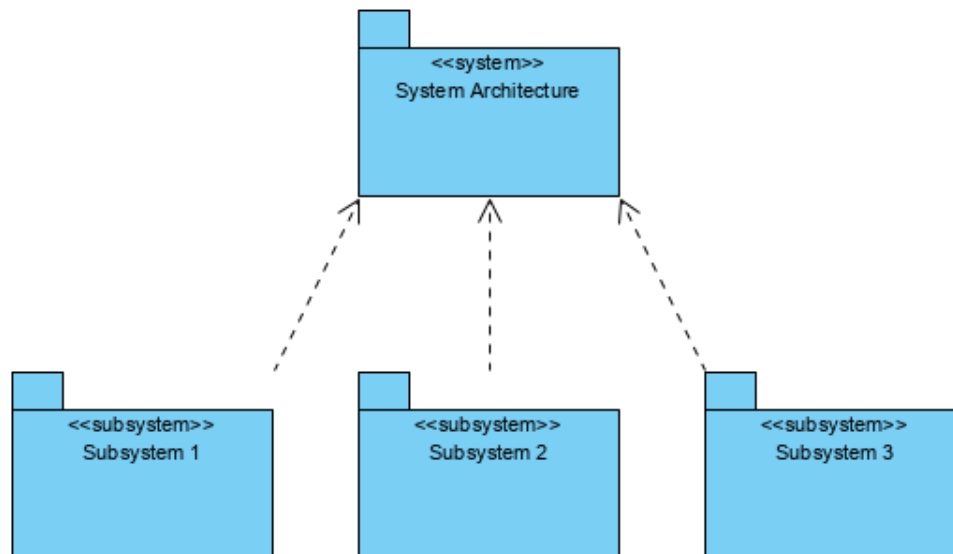
- Tienen un objetivo común o relaciones conceptuales fuertes,
- Pertenecen a un mismo árbol de herencia, o
- Pertenecen al mismo caso de uso.

O formando paquetes mas grandes de tipo arquitectural:

- Paquete "Clases e interfaces del modelo".
- Paquete "Interfaces de usuario".
- Paquete "Servicios base de datos".

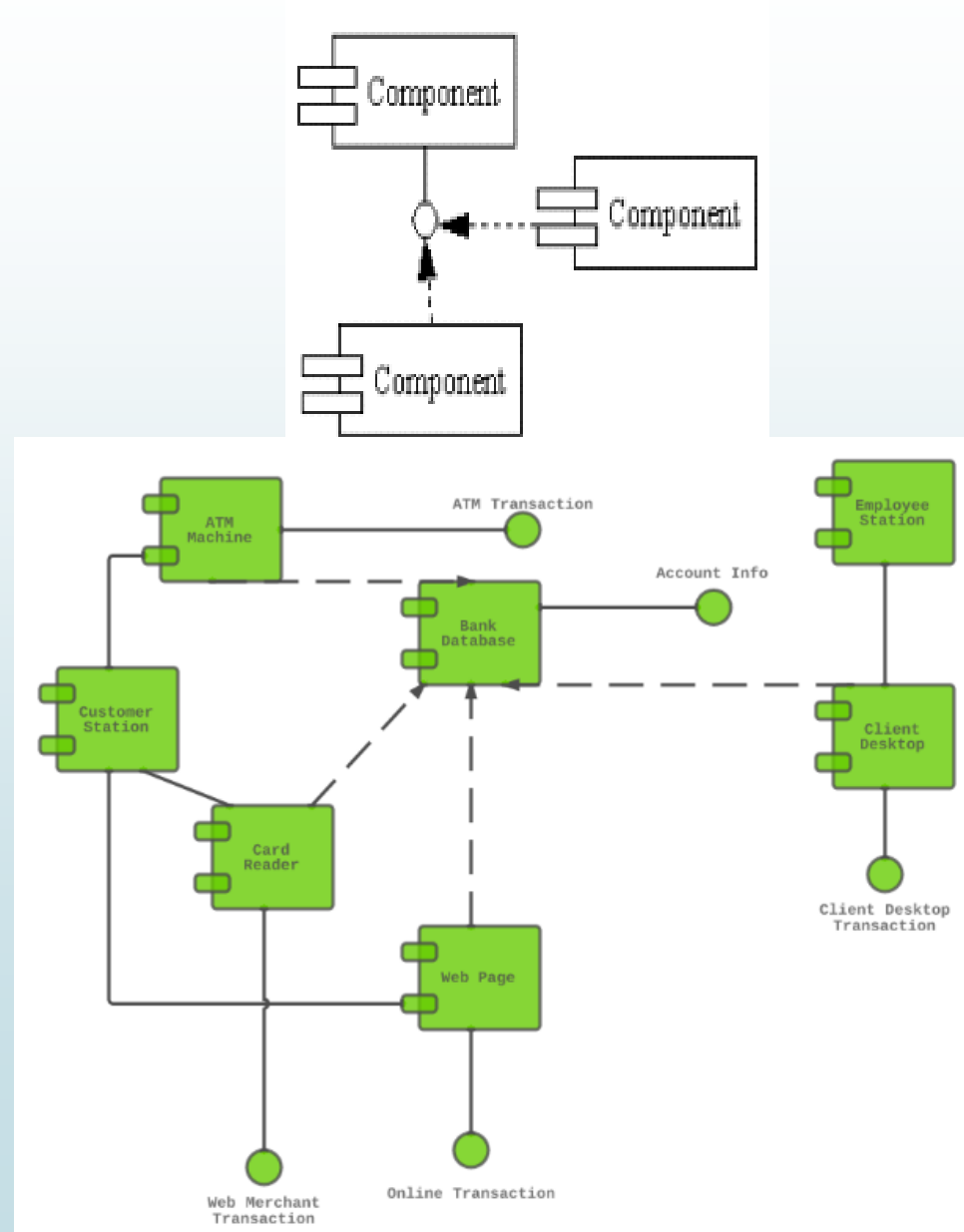
# Diagrama de Subsistemas

- El sistema representa todos los elementos del modelo que pertenecen al proyecto en particular.
- Un sistema puede descomponerse recursivamente en subsistemas más pequeños
- Un subsistema es una agrupación de elementos, algunos de los cuales constituyen una especificación del comportamiento ofrecido por otros elementos.
- Elemento de modelado que tiene semántica package+clase que provee comportamiento
- El uso de subsistemas fuerza a la modularidad & encapsulación
- Un sistema se representa como un paquete con el estereotipo de <<sistema>>








# Diagrama de componentes

- Describe la organización de los componentes físicos del sistema
- Un componente es la realización física de una abstracción en el diseño
- Los componentes corresponden a la implementación
- Ejemplos de componentes son:
  - .exe, .dll, .java, .class
- Se pueden estereotipar como: <<source code>>, <<file>>, <<executable>>, entre otros.

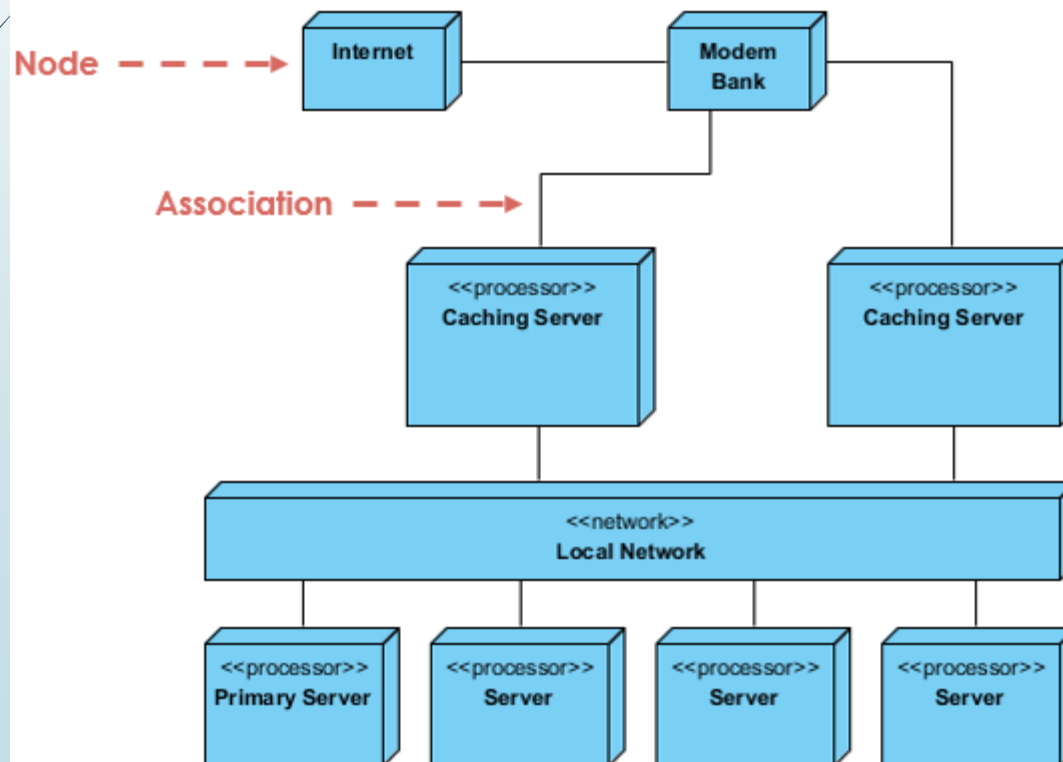


## Notación utilizada en diagrama de componentes

Elemento	Símbolo/notación	Explicación
Componente ( <i>component</i> )		Símbolo para representar los módulos de un sistema (la interacción y la comunicación tienen lugar a través de interfaces).
Interfaz ofrecida		Símbolo para una o más interfaces claramente definidas que proporcionan funciones, servicios o datos al mundo exterior (el semicírculo abierto también se denomina enchufe o <i>socket</i> ).
Interfaz requerida		Símbolo de una interfaz necesaria para recibir funciones, servicios o datos del exterior (la notación del círculo con palo también se denomina <i>lollipop</i> o piruleta).
Relación		Las líneas actúan como conectores e indican las relaciones entre los componentes.
Relación de dependencia		Conector especial para expresar una relación de dependencia entre los componentes del sistema (no siempre se indica).

# Diagrama de Deployment o despliegue

- Un diagrama de implementación UML es un diagrama que muestra la configuración de los nodos de procesamiento de tiempo de ejecución y los componentes que viven en ellos.
- Los diagramas de implementación son un tipo de diagrama de estructura utilizado para modelar los aspectos físicos de un sistema orientado a objetos.
- Se usan para modelar la vista de implementación estática de un sistema (topología del hardware).



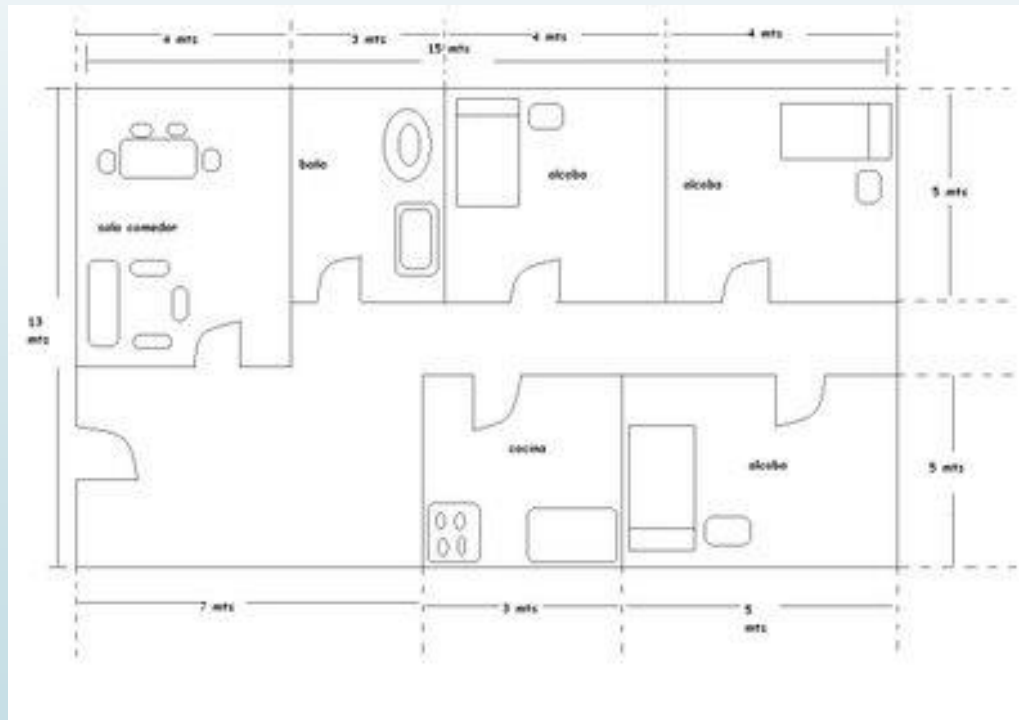
# Modelo “4+1” vistas de Kruchten

- El modelo “4+1” de Kruchten, es un modelo de vistas diseñado por el profesor [Philippe Kruchten](#) y que encaja con el estándar “IEEE 1471-2000” que se utiliza para describir la arquitectura de un sistema software basado en el uso de múltiples puntos de vista.



# Vista

- **Una vista** es una representación de todo el sistema software desde una determinada perspectiva, y **un punto de vista** se define como un conjunto de reglas (o normas) para realizar y entender las vistas.

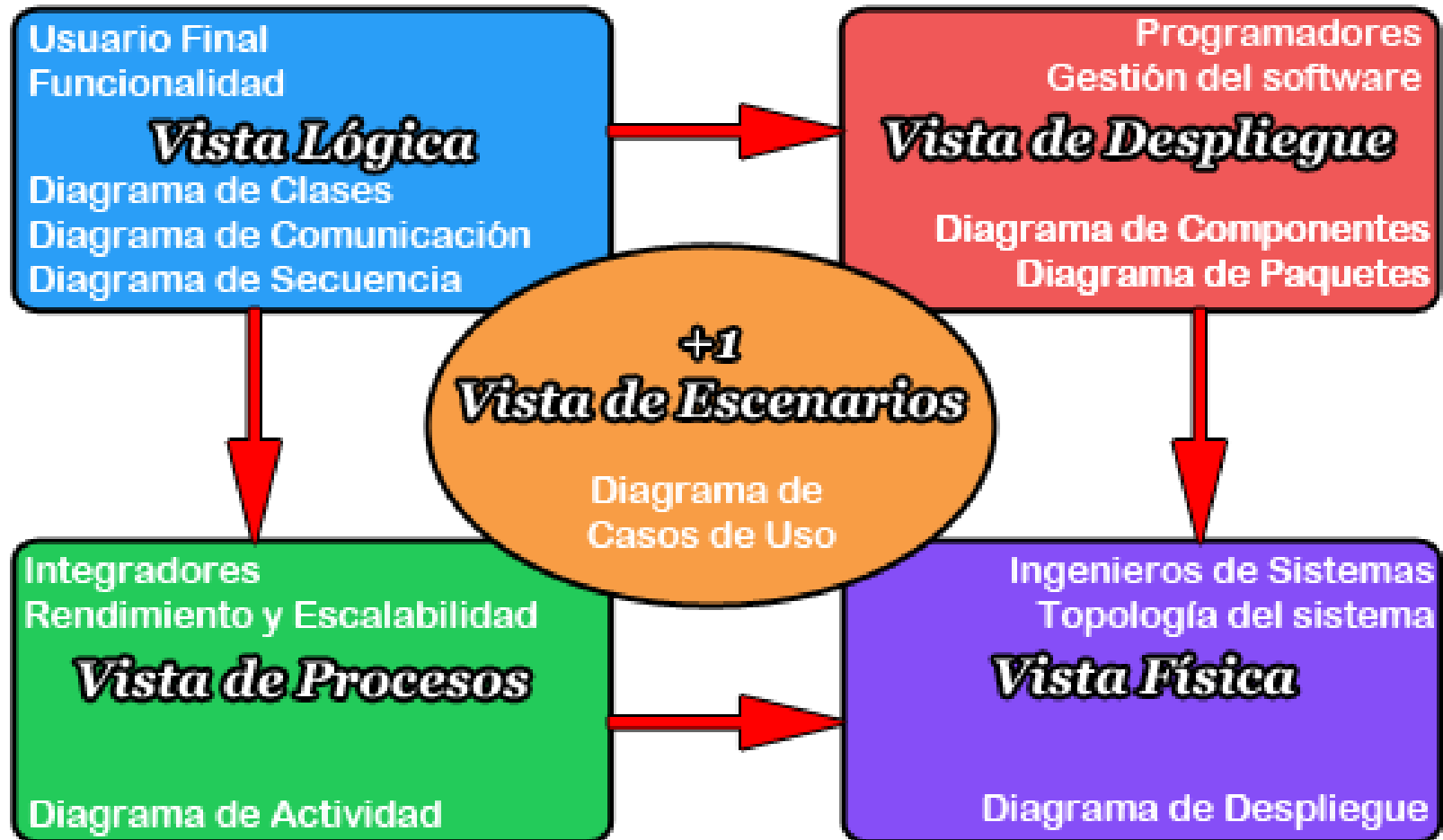


# Vistas 4+1

- Lo que propone Kruchten es que un sistema software se ha de documentar y mostrar (tal y como se propone en el estándar IEEE 1471-2000) con 4 vistas bien diferenciadas y estas 4 vistas se han de relacionar entre sí con una vista más, que es la denominada vista “+1”. Estas 4 vista las denominó Kruchten como:
  - **Vista lógica,**
  - **Vista de procesos,**
  - **Vista de despliegue**
  - **Vista física y**
  - **Vista “+1”** (que tiene la función de relacionar las 4 vistas citadas, la denominó vista de escenario)
- Cada una de estas vistas ha de mostrar toda la arquitectura del sistema software que se esté documentando, pero cada una de ellas ha de documentarse de forma diferente y ha de mostrar aspectos diferentes del sistema software

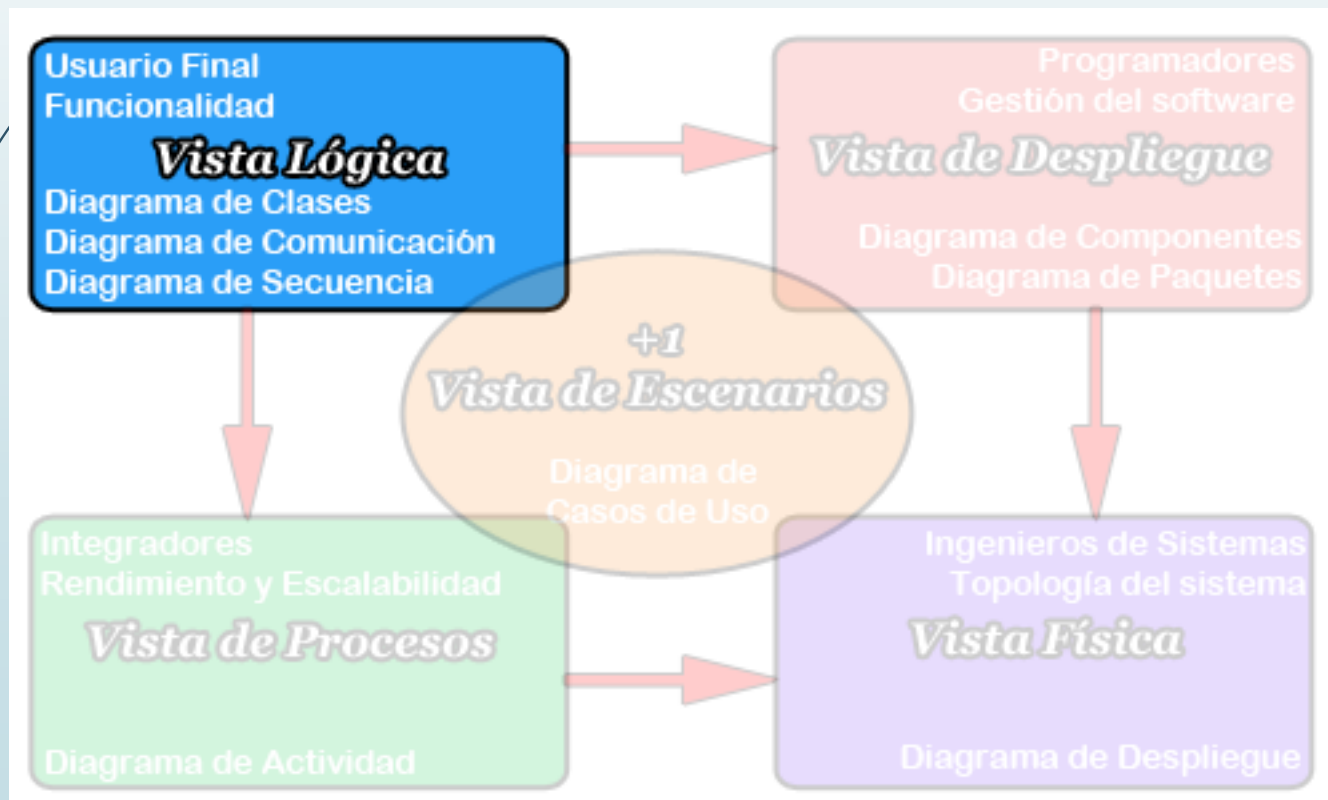


# Vistas 4 +1



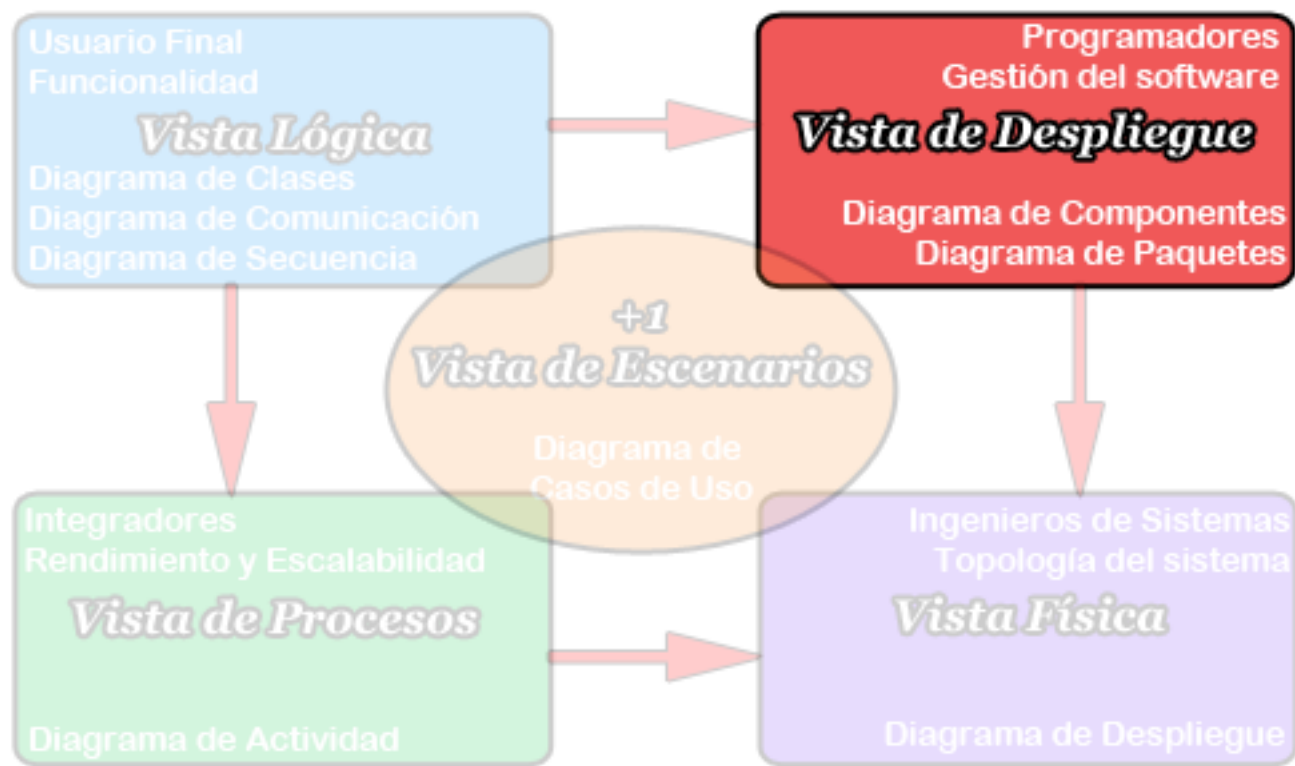
# Vista Lógica

- En esta vista se representa la funcionalidad que el sistema proporcionara a los **usuarios finales**. Es decir, se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece. Para completar la documentación de esta vista se pueden incluir los diagramas de clases, de comunicación o de secuencia de UML.



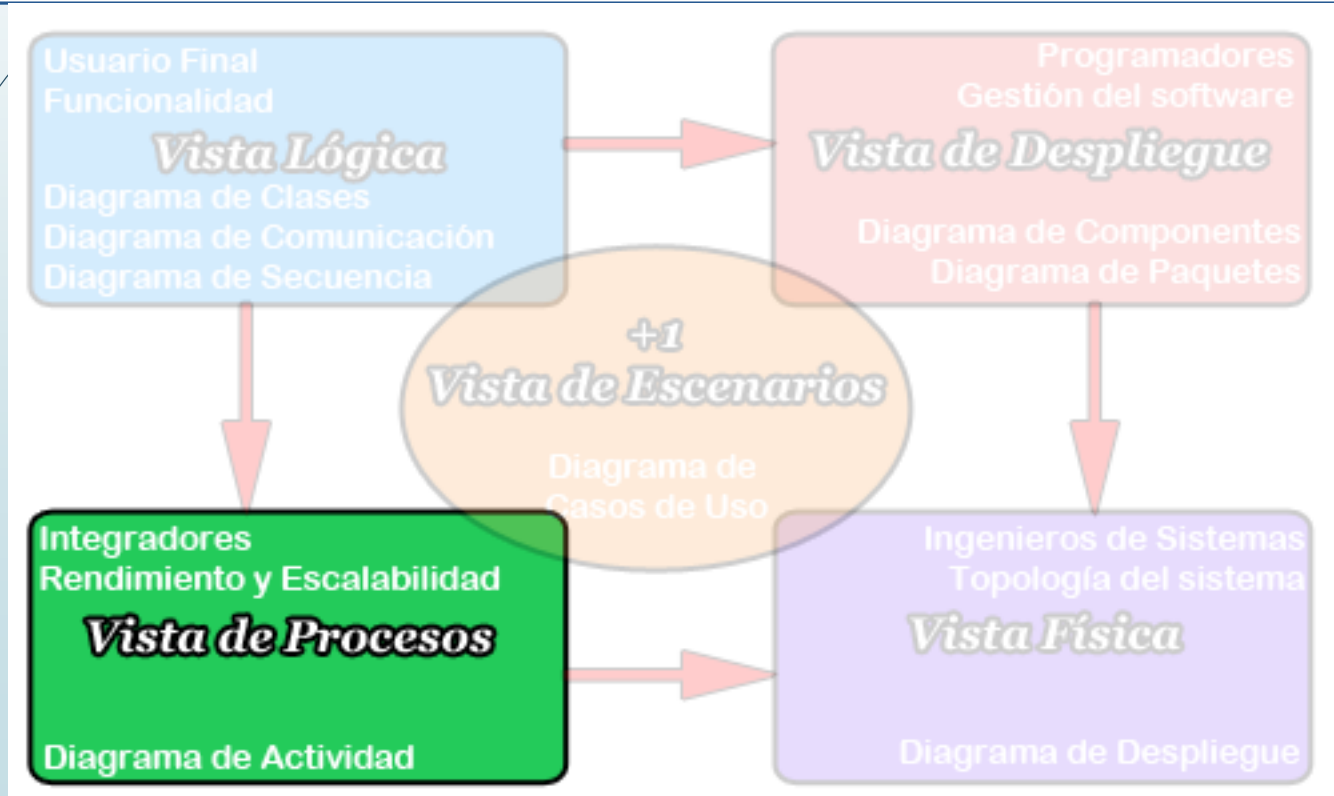
# Vista de despliegue

- En esta vista se muestra el sistema desde la perspectiva de **un programador** y se ocupa de la gestión del software; o en otras palabras, se va a mostrar como esta dividido el sistema software en componentes y las dependencias que hay entre esos componentes. Para completar la documentación de esta vista se pueden incluir los diagramas de componentes y de paquetes de UML.



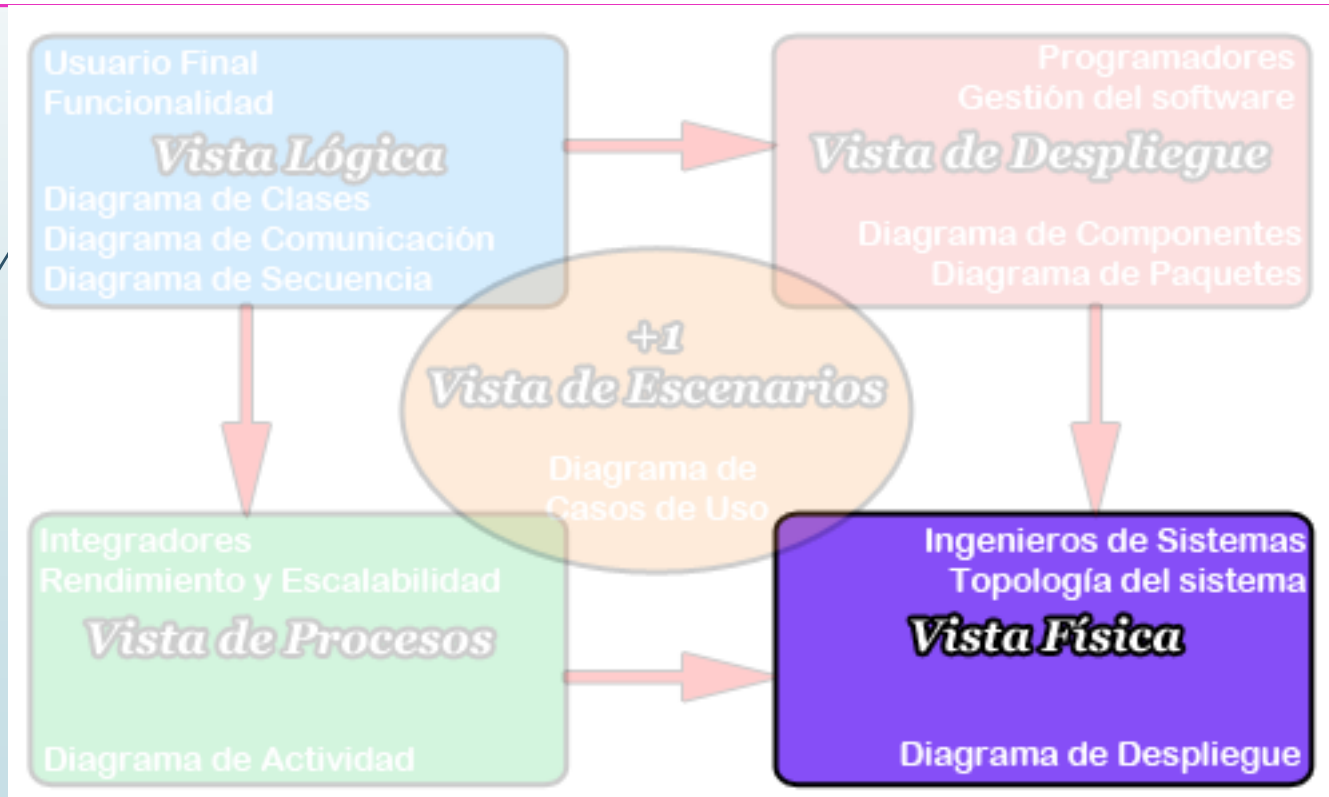
# Vista de Procesos

- En esta vista se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos; es decir, se representa desde la perspectiva de un **integrador de sistemas**, el flujo de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema. Para completar la documentación de esta vista se puede incluir el diagrama de actividad de UML.



# Vista Física

- En esta vista se muestra desde la perspectiva de **un ingeniero de sistemas** todos los componentes físicos del sistema así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios). Para completar la documentación de esta vista se puede incluir el diagrama de despliegue de UML.



# “+1” Vista de Escenarios

- Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas, esto quiere decir que desde un caso de uso podemos ver como se van ligando las otras 4 vistas, con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc., para realizar cada caso de uso. Para completar la documentación de esta vista se pueden incluir el diagramas de casos de uso de UML.

