Diseño de Sistemas Software

PEEA

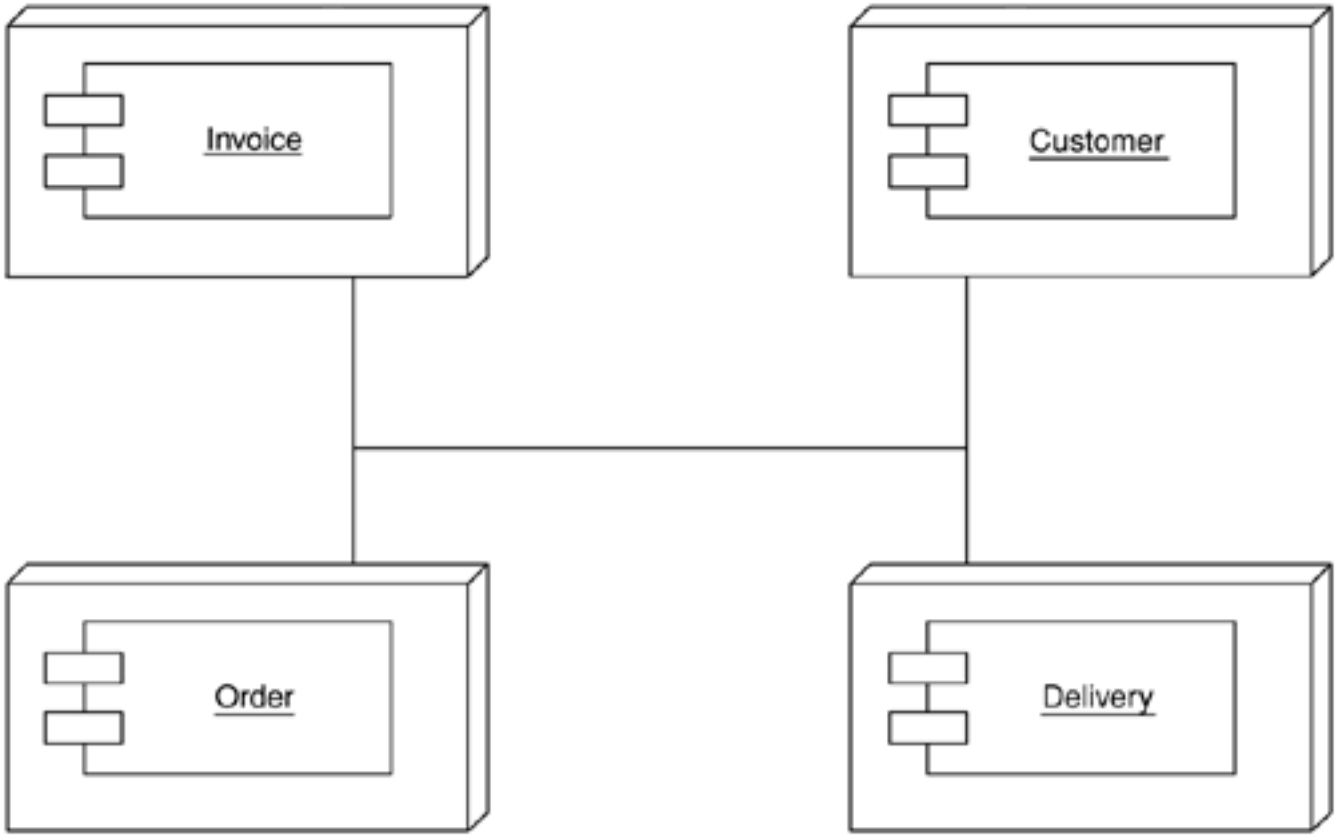
Estrategias de distribución

Introducción

- Un **sistema distribuido** es aquél que se ejecuta repartido en distintos procesos o máquinas
- Realizar un **diseño distribuido de objetos** implica ubicar los objetos del sistema en los diferentes nodos donde se ejecutarán

Introducción

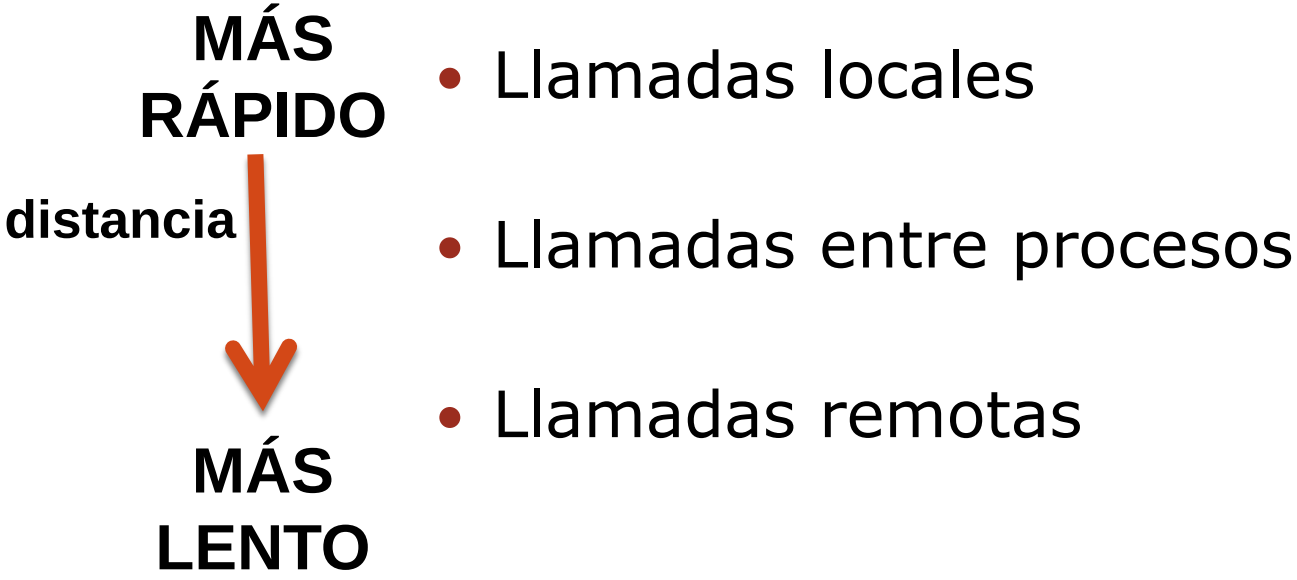
- Cuál es el problema de este diseño?



Aplicación distribuida con diferentes componentes en distintas máquinas

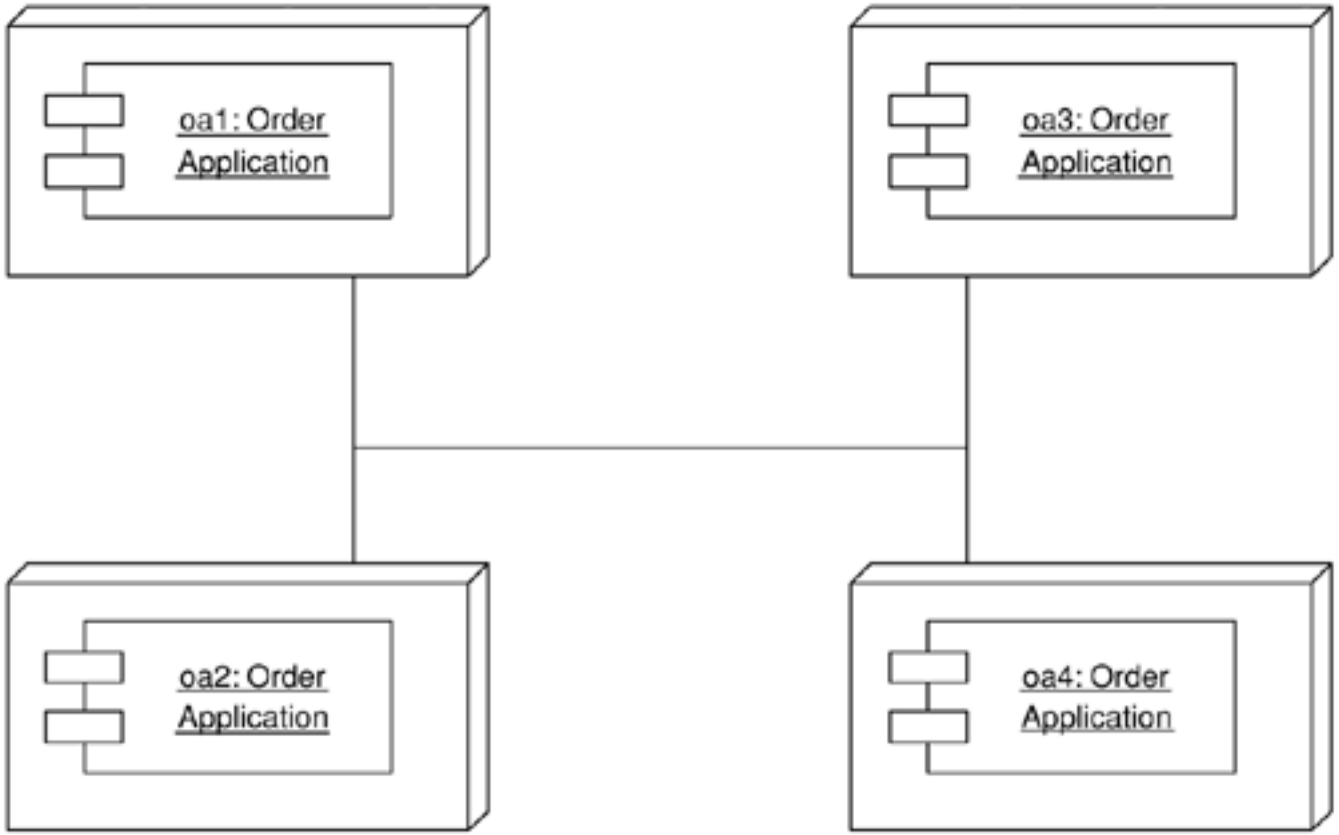
Introducción

- Al distribuir objetos tenemos que tener en cuenta el rendimiento del sistema
- La comunicación entre procesos o entre sistemas remotos implica una disminución del rendimiento



Clustering

- Mejora: poner varias copias de la aplicación completa en distintos nodos (**clustering**)



Estrategias de coordinación

Primera Ley del

Diseño de Objetos Distribuídos:

"¡No distribuyas los objetos!"

Estrategias de distribución

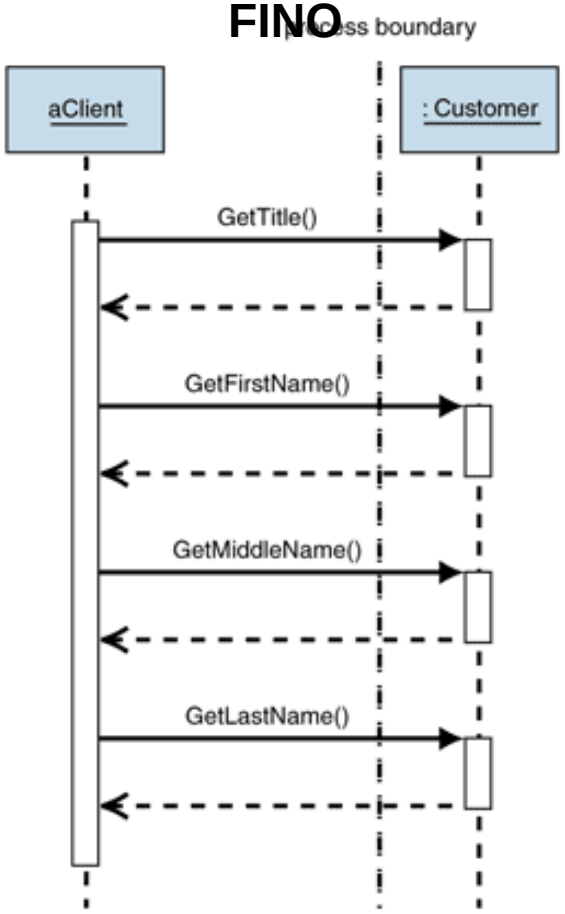
- Desafortunadamente, hay algunas situaciones en las que no se puede evitar la distribución de objetos
  - Sistemas cliente-servidor
  - Servidor de aplicación y base de datos (comunicación SQL)
  - Servidor web y servidor de aplicación
  - Sistemas de terceros que necesitan funcionar en su propio proceso
  - **Situaciones imprevistas que puedan aplicarse a vuestros proyectos**

Estrategias de distribución

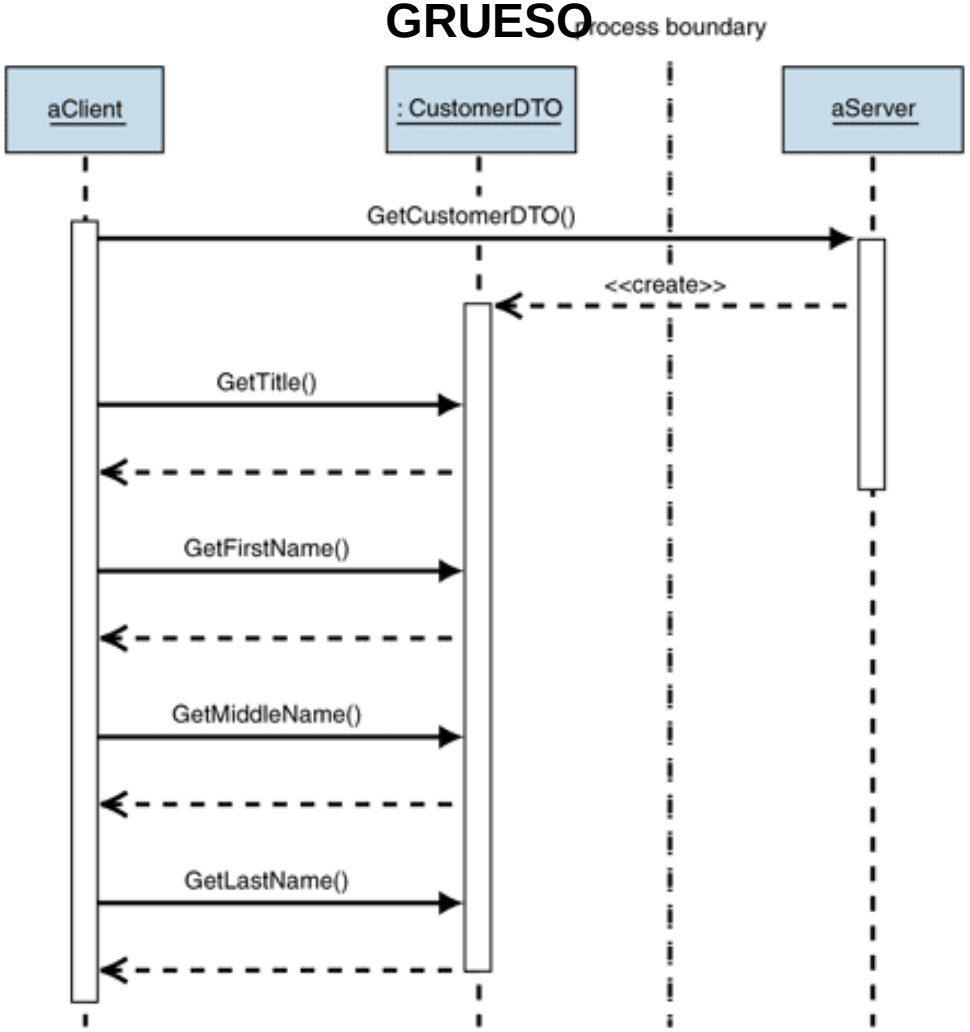
- Qué hacer cuando tienes que distribuir:
  - Usar **interfaces de grano fino** para los objetos locales (como de costumbre), permite hacer un mejor diseño OO
  - Usar **interfaces de grano grueso** para acceder a los objetos remotos para disminuir en la medida de lo posible la pérdida de rendimiento que implica hacer llamadas remotas

Estrategias de distribución

INTERFAZ DE GRANO FINO



INTERFAZ DE GRANO GRUESO



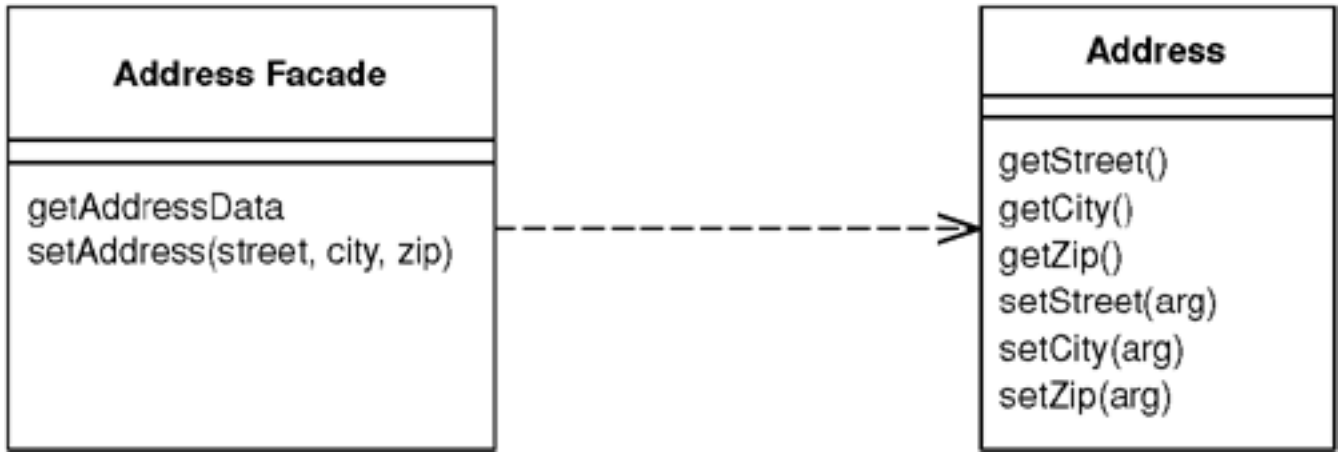
Estrategias de distribución

- Remote Façade
- Data Transfer Object

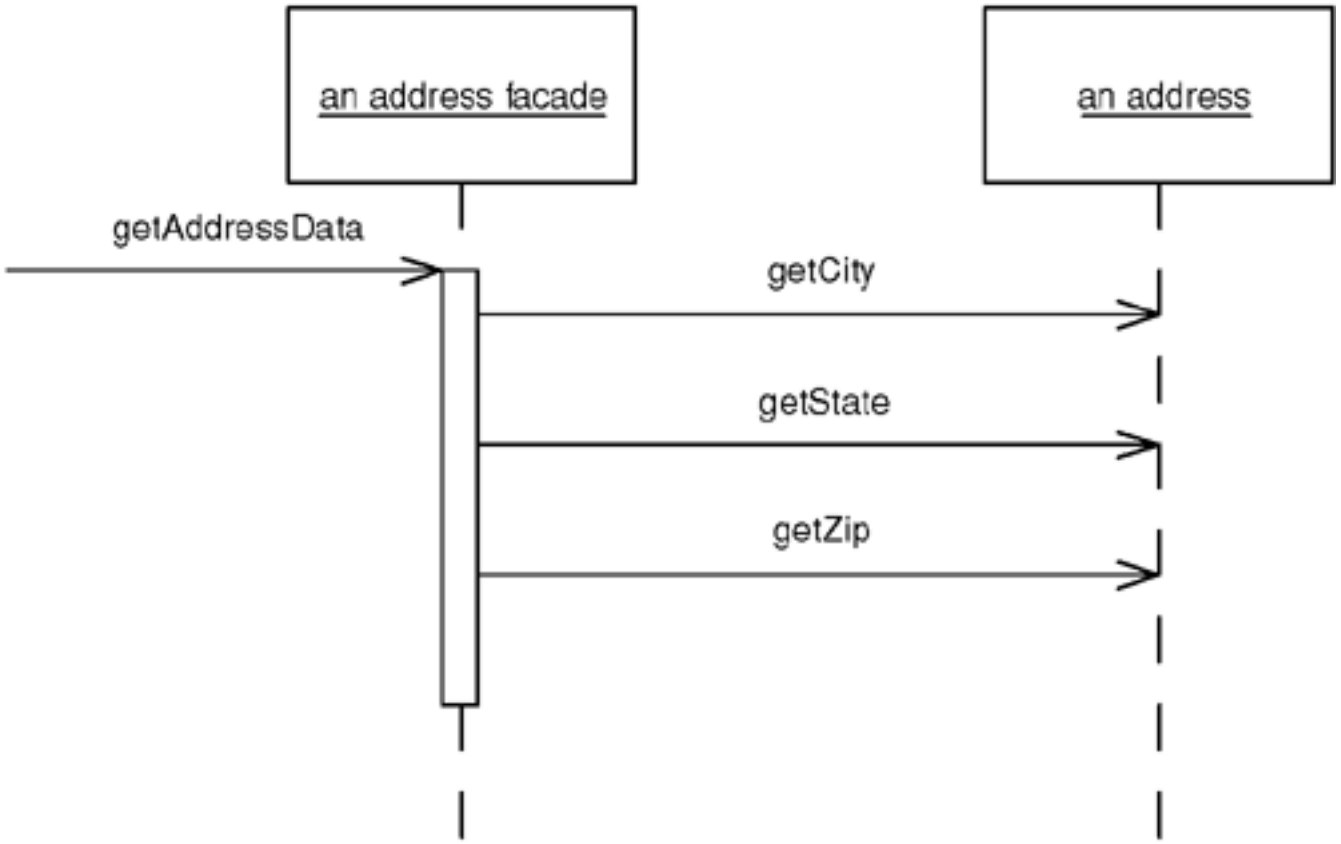
Remote Façade

- Definición:

  "Provee una fachada de grano grueso para acceder a objetos de grano fino, para mejorar el rendimiento a través de la red."



Remote Façade



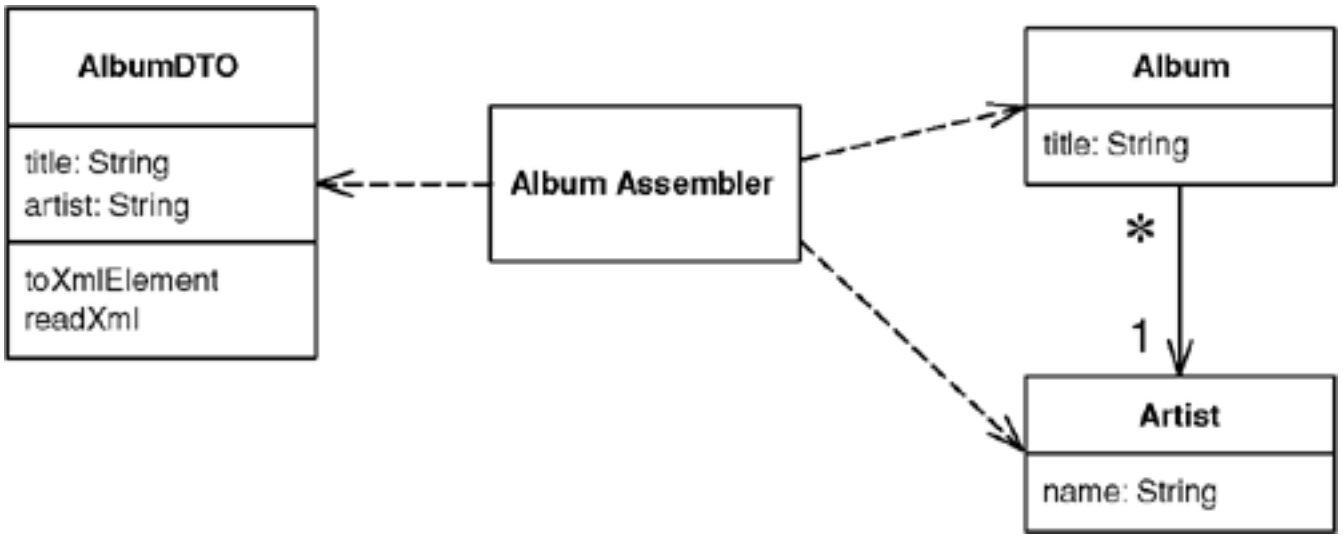
Remote Façade

- Una **Fachada Remota** reemplaza todos los métodos get() y set() con un único método para acceder a todas las propiedades del objeto
- Provee un interfaz de grano grueso sin modificar los objetos de dominio
- **Las Fachadas Remotas no contienen lógica de negocio!**
- Se puede usar una única Fachada Remota para acceder a varios objetos de dominio
- Las Fachadas Remotas pueden implementerse con estado (p.ej. SOAP) o sin estado (p.ej. REST)

Data Transfer Object

- Definición:

  "Un objeto que transporta datos entre procesos para reducir el número de llamadas."

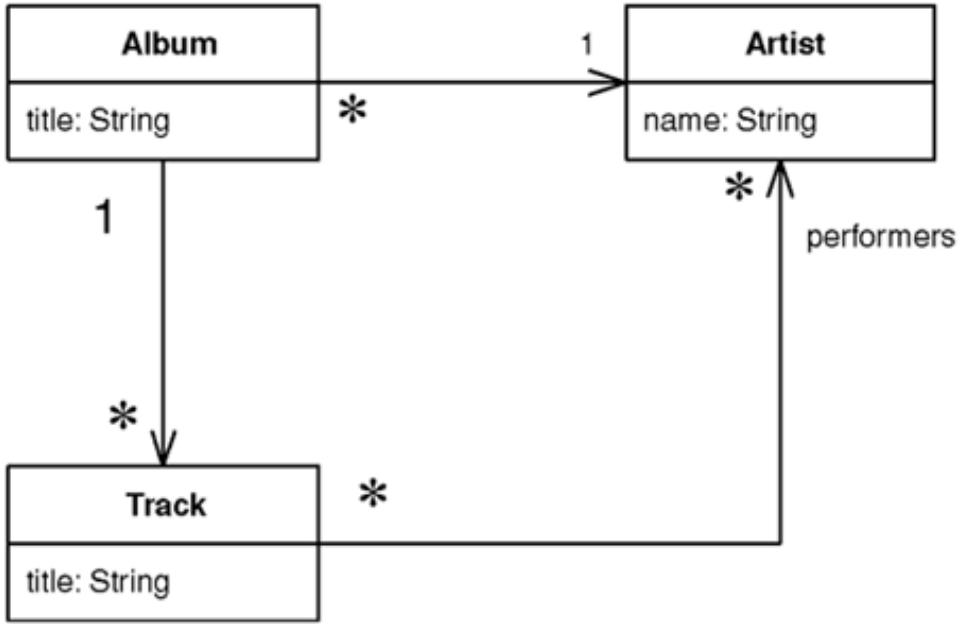


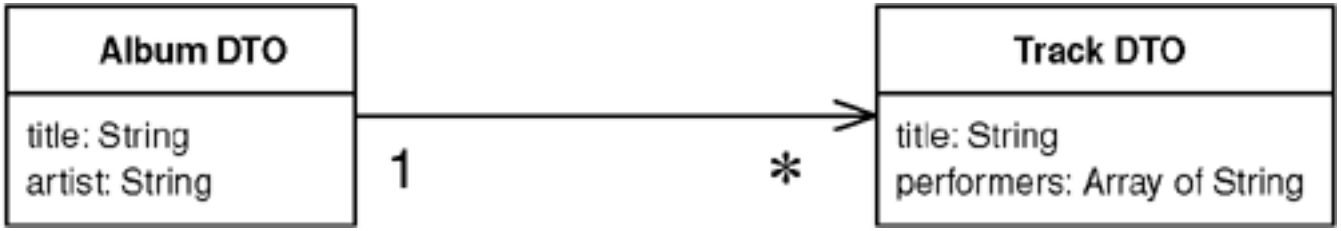
Data Transfer Object

- Contiene múltiples datos para transportarlos en una única llamada remota
- Debe ser *serializable* para poder transferir los datos sobre la conexión
  - Formato binario: es más compacto pero más sensible a errores (p.ej. clientes no actualizados)
  - Formato textual: (p.ej. XML) necesita más ancho de banda pero es más robusto ante los cambios

Data Transfer Object

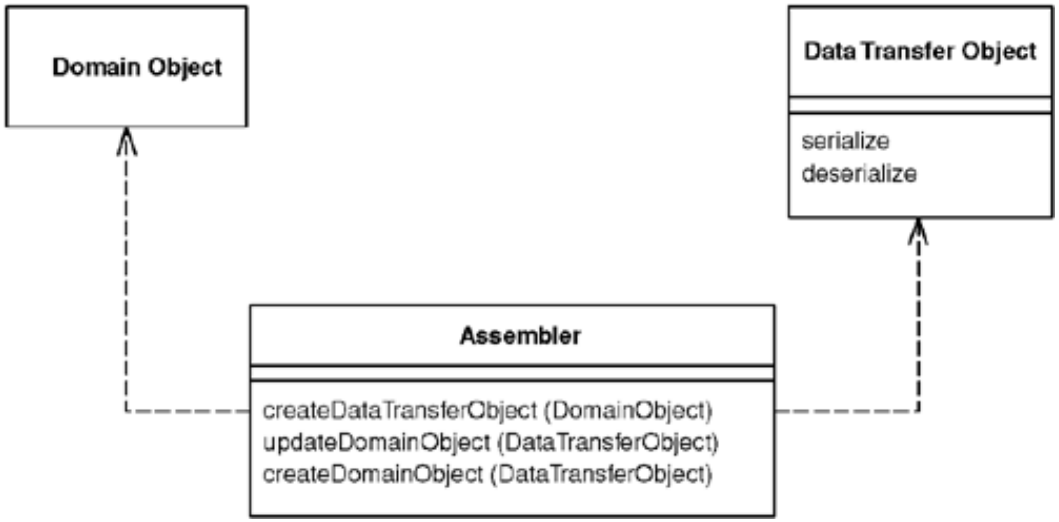
- Los DTOS normalmente contienen datos de varios objetos de dominio





Data Transfer Object

- Los DTO y los objetos de dominio deben estar desacoplados (los DTO deben desplegarse en los dos lados, pero los objetos de dominio no)
- Se puede usar un ensamblador para construir DTOs a partir de los objetos de dominio



Bibliografía

- Fowler, Martin

  Patterns of Enterprise Application Architecture

  Addison Wesley, 2003.