Patrones de Diseño: Patrones Estructurales. Tema 4-8: Proxy

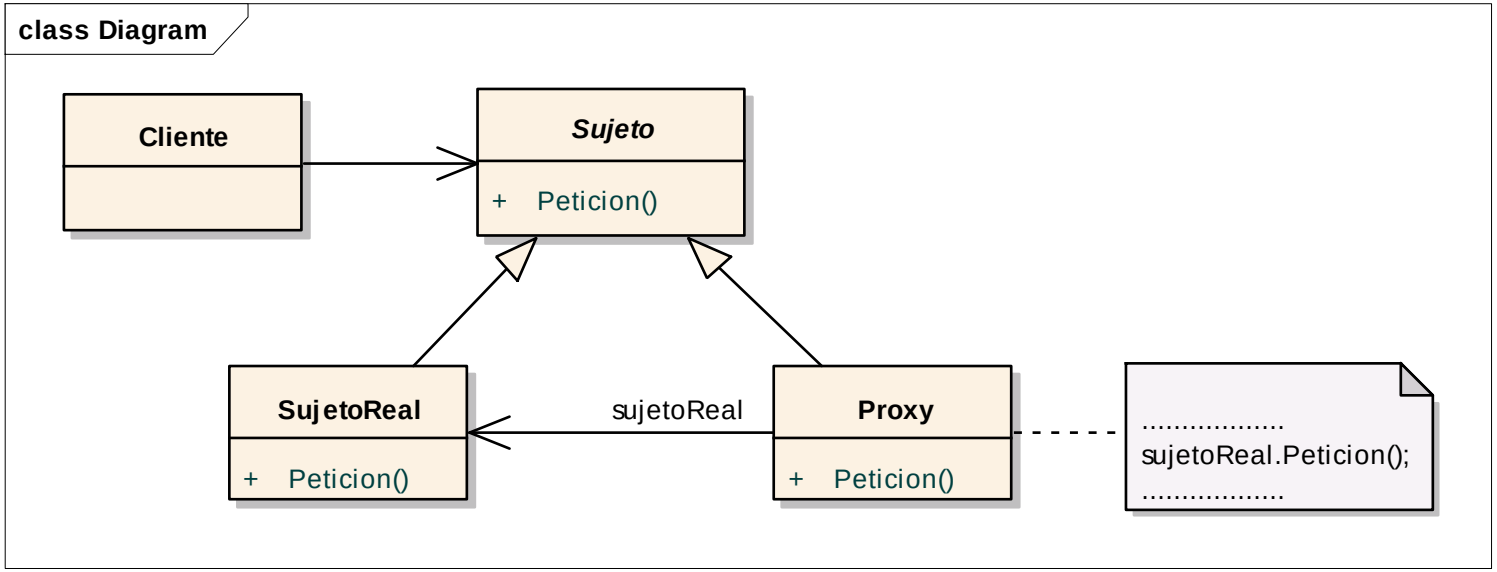
Descripción del patrón

- **Nombre:**
  - Representante
  - También conocido como Virtual Proxy o Surrogate
- **Propiedades:**
  - Tipo: estructural
  - Nivel: objeto, componente
- **Objetivo o Propósito:**
  - Proporciona un representante o sustituto de otro objeto para controlar el acceso a éste.

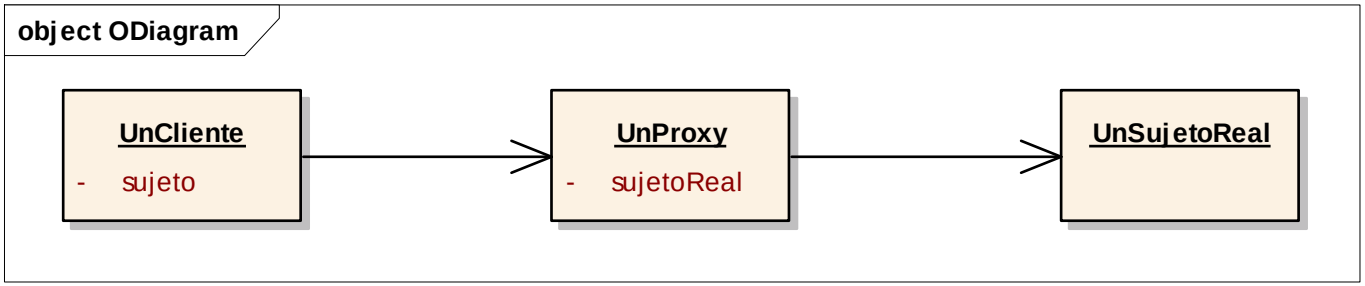
Aplicabilidad

- Use el patrón Proxy cuando necesite sustituir una referencia simple a un objeto por una referencia más elaborada. Usos:
  - **Proxy Remoto:** Cuando el objeto está en un sistema remoto y necesite un representante local.
  - **Proxy Virtual:** Para retrasar la creación de objetos costosos hasta que sean necesarios.
  - **Proxy de Protección:** Para controlar los derechos de acceso a un objeto.
  - **Proxy de Sincronización:** Para gestionar accesos de múltiples clientes a un recurso.
  - **Referencias Inteligentes:** Para gestión y mantenimiento del acceso a un objeto real, permite contabilizar su utilización e incluso destruirlo cuando ya no se necesita. También permite sincronizar accesos concurrentes al mismo, bloqueándolo mientras está en uso.

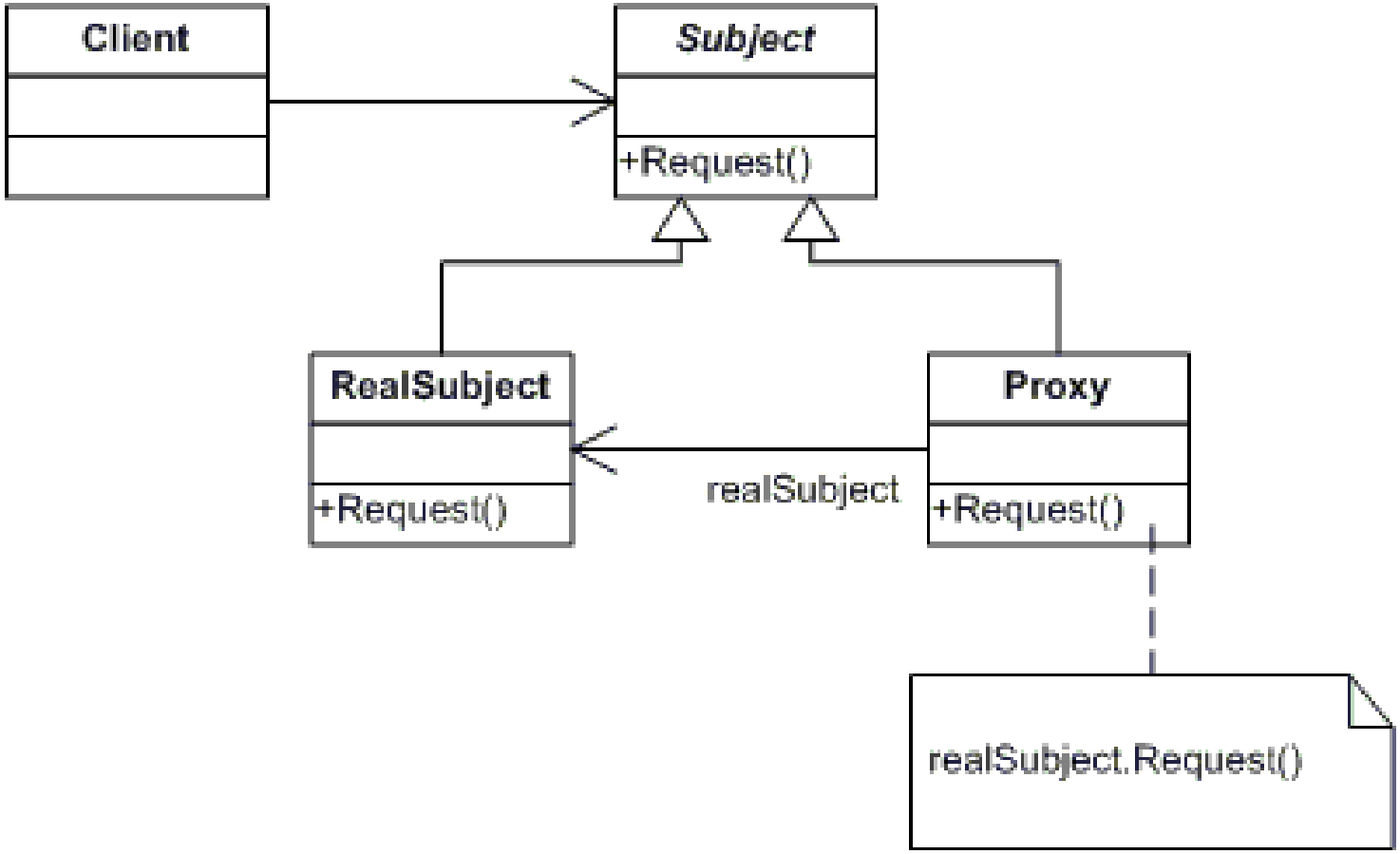
Estructura



Este es un posible diagrama de objetos de una estructura de proxies en tiempo de ejecución:



Estructura


Client
Subject
+Request()
RealSubject
+Request()
realSubject
Proxy
+Request()
realSubject.Request()

Estructura. Participantes

- **Sujeto:** Define la interfaz común para el Proxy y el SujetoReal de forma que se pueda usar un Proxy donde se espere un SujetoReal.
- **Proxy:** Ofrece una interfaz equivalente al de la clase SujetoReal, y redirige las llamadas de los métodos al objeto real. Puede realizar un pre-procesamiento y un post-procesamiento sobre los servicios ofrecidos por la clase real.
- **SujetoReal:** Es la clase que implementa los servicios reales ofrecidos, puede ser una instancia local o remota.
- **Cliente:** La aplicación, utiliza la interfaz de la clase proxy para hacer uso de la clase real.

Consecuencias

- El patrón Proxy introduce un nivel de indirección al acceder a un objeto.
- Puede mejorar la eficiencia al retrasar la instanciación de un objeto costoso hasta que sea necesario utilizarlo. Así el objeto proxy lo sustituye ofreciendo la misma interfaz, solo cuando es necesario le solicita al objeto real la información que necesita.
- Aumenta la seguridad.
- Los clientes se desentienden de la ubicación de los componentes accedidos.

Implementación

- Se diseña la clase Proxy con igual interfaz que la clase real, si es posible se hereda de una clase abstracta común a la clase real (o se implementa una interfaz).
- Según el tipo de proxy (remoto, virtual, etc.) se le añaden métodos de pre-procesamiento y post-procesamiento donde se implementan las funciones de control, etc.
- Se enlaza el Proxy con la clase real mediante instancias locales, referencias o punteros, sockets, etc.
- Se implementa el cliente haciendo uso de la clase Proxy en lugar de la real.

Patrones relacionados

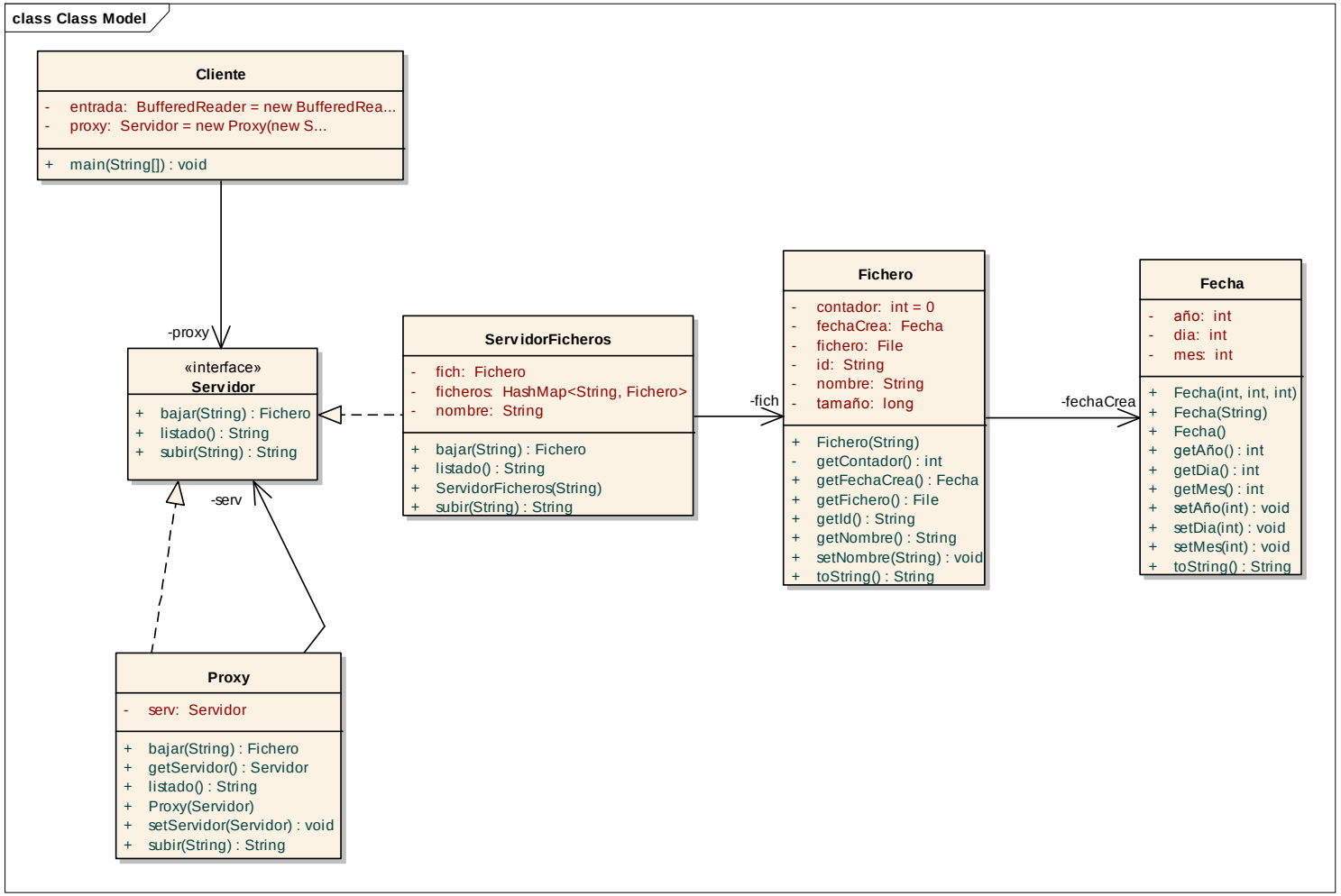
- **Adapter:** Un adaptador proporciona una interfaz diferente para el objeto que adapta. Por el contrario un Proxy tiene la misma interfaz que su objeto real.

- **Decorator:** Aunque la implementación de ambos patrones es parecida, un decorador añade una o más responsabilidades a un objeto, mientras que un Proxy controla el acceso al mismo.

Código de ejemplo

SERVIDORES

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Sujeto: Servidor.
  - Proxy:Proxy.
  - SujetoReal: ServidorFicheros.
  - Cliente: Cliente.