POLITÉCNICA

"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

MiW

Patrones de Diseño

18. *Mediator*

Luis Fernández Muñoz

https://www.linkedin.com/in/luisfernandezmunyoz

setillofm@gmail.com

INDICE

1. Problema

- *En un partido de futbol de una liga profesional hay mucho más que dos equipos de jugadores y un árbitro; se contempla el público, los cuerpos de seguridad, el delegado de campo, …*
- *Para evitar que todos estos agentes interactúen con todos los otros (p.e. un policía ordenando retirarse a un defensa por seguridad, un delegado de campo solicitando una prorroga para el comienzo a todos los jugadores, …) en un gran caos de sincronización, se establece la figura del árbitro al que todos acuden y éste actúa sobre todos según considere oportuno.*

- *Un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve bajo acoplamiento evitando a los objetos referirse explícitamente a otros y permite variar su interacción de forma independiente.*

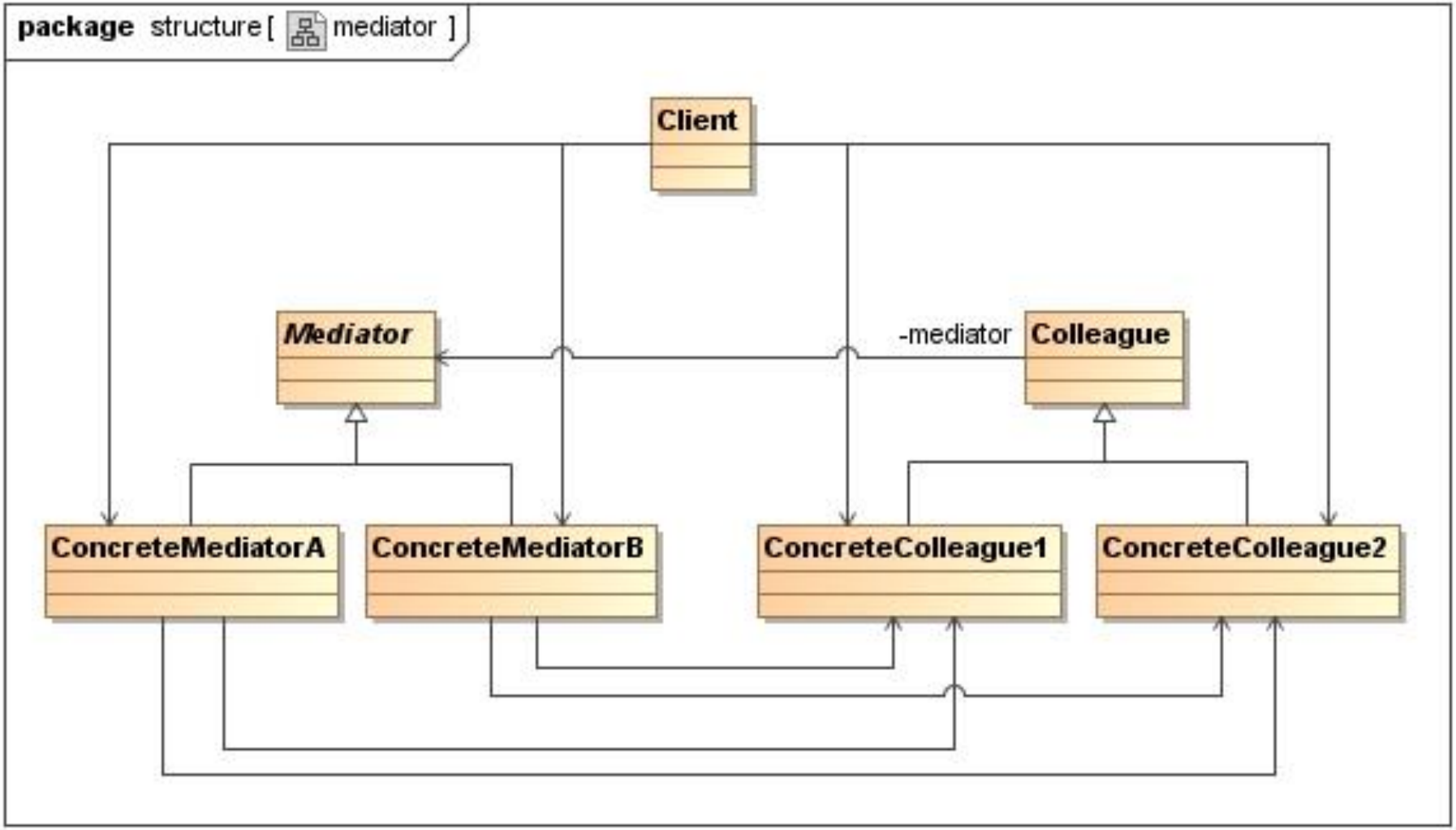
1. Problema

- La distribución de comportamientos entre objetos puede resultar en una estructura de objetos con muchas conexiones entre ellos; en el peor caso, cada objeto conoce a cada uno de los otros
  - Montones de conexiones hace menos possible que un objeto pueda trabajar sin el soporte de los otros – el sistema actúa como si fuera un monolito
  - Esto dificulta cambiar de cualquier forma el comportamiento del sistema, dado que el comportamiento está distribuido entre muchos objetos. Como resultado, se podría forzar a definir muchas subclases para particularizar el sistema.

1. Problema

- Usa el patron Medidor cuando:
  - Un conjunto de objetos se comunican bien definidos pero de forma compleja. El resultado de las interdependencies no están estructuradas y son difíciles de entender.
  - Reusar un objeto es difícil porque se refiere y se comunica con muchos otros objetos
  - Un comportamiento distribuido entre varias clases debería ser particularizado sin muchas subclasificaciones

2. Solución


package structure [mediator]
Client
Mediator
-mediator
Colleague
ConcreteMediatorA
ConcreteMediatorB
ConcreteColleague1
ConcreteColleague2

2. Solución

- No se necesita definir una clase abstracta Mediador cuando los colegas trabajan solo con un mediador. Lo que permite es trabajar a los colegas con diferentes subclases y viceversa
- Define una interfaz de notificación en el Mediador que permite a los colegas ser más directos en la comunicación. Cuando se comunican con el Mediador, un colega se pasa como argumento, permitiendo al Mediador identificar al emisor

3. Consecuencias

- Promueve el desacoplamiento entre colegas. Se pueden variar y reutilizar las clases colega y mediador independiente.
- Localiza el comportamiento que de otro modo se distribuiría entre varios objetos. El cambio de este comportamiento requiere únicamente subclases de mediador. Las clases colegas pueden reutilizarse como están.
- Sustituye a interacciones muchos-a-muchos con interacciones de uno a muchos entre el mediador y sus colegas. Las relaciones uno-a-muchos, son más fáciles de entender, mantener y extender.
- Permite centrarse en cómo los objetos interactúan aparte de su comportamiento individual. Eso puede ayudar a aclarar cómo los objetos interactúan en un sistema.
- Debido a que encapsula protocolos, puede ser más complejo que cualquier colega individual. Esto puede hacer que el mediador sea un monolito que es difícil de mantener.

4. Relaciones

- *Observer* para comunicar los colegas con el *Mediator*

5. Comparativa

- ***Mediator*** *vs* ***Observer***
  - Son rivales y la diferencia es que *Observer* distribuye la comunicación, mientras que *Mediator* encapsula la comunicación entre objetos
    - *Mediator* centraliza más que distribuye, ubicando en el *Mediator* la responsabilidad de mantener la restricción.
    - *Observer* no tiene ningún objeto individual que encapsule una restricción. En vez de eso, el *Observer* y el Sujeto deben cooperar para mantener la restricción. Las vías de comunicación se determinan por el modo en que se interconectan: un sujeto individual normalmente tiene muchos observadores, y a veces el observador de un sujeto es un sujeto de otro observador. El patrón *Observer* promueve la separación y bajo acoplamiento, lo que conduce a clases de grano más fino que son más aptas para ser reutilizadas
    - Es más fácil hacer *Observer* reutilizables que hacer *Mediator* reutilizables. Pero es más fácil entender el flujo de comunicación en el *Mediator* que en el *Observer* .

5. Comparativa

- ***Mediator** vs **Facade***
  - Ver *Facade vs Mediator*