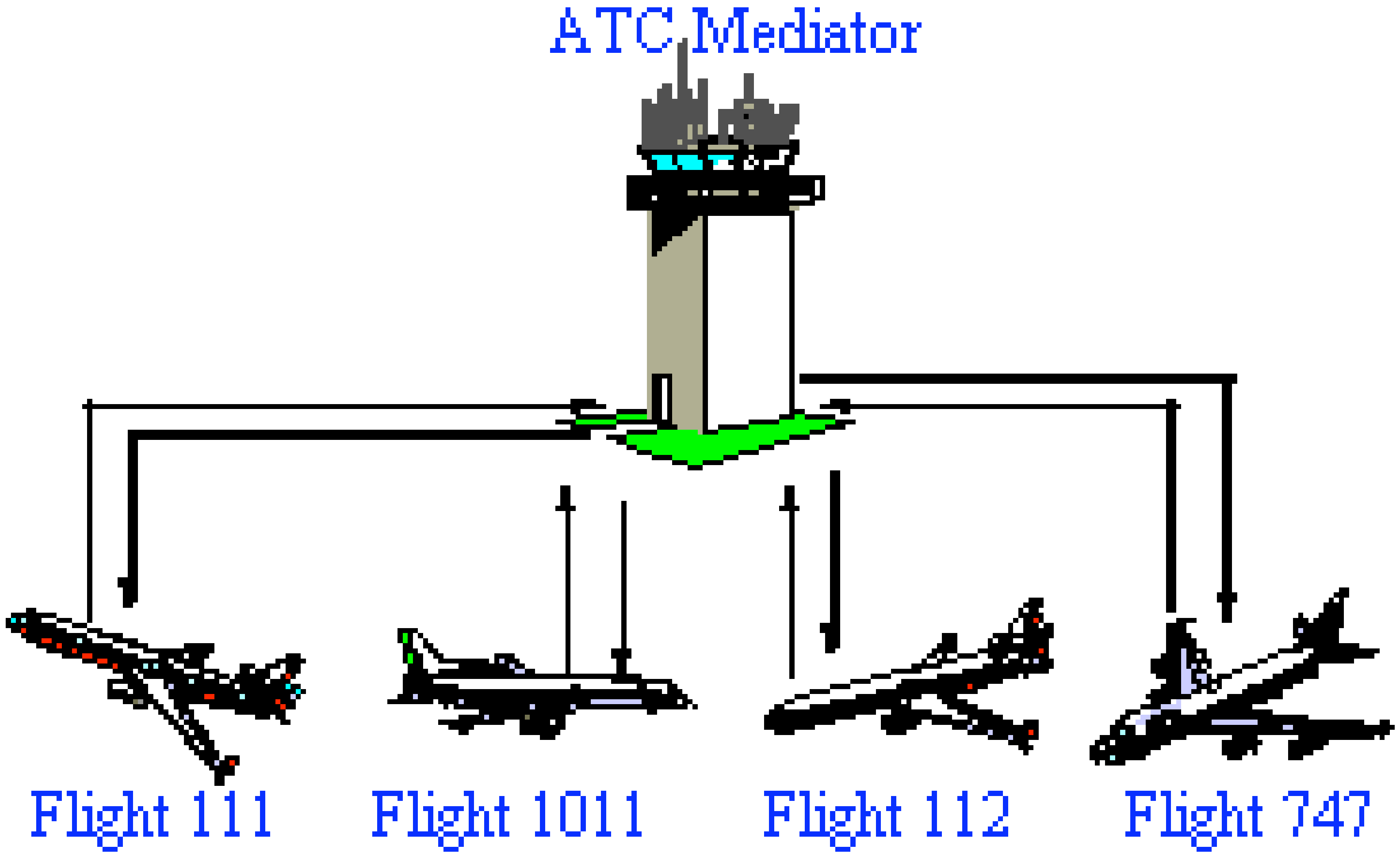
Tecniche di Progettazione: Design Patterns

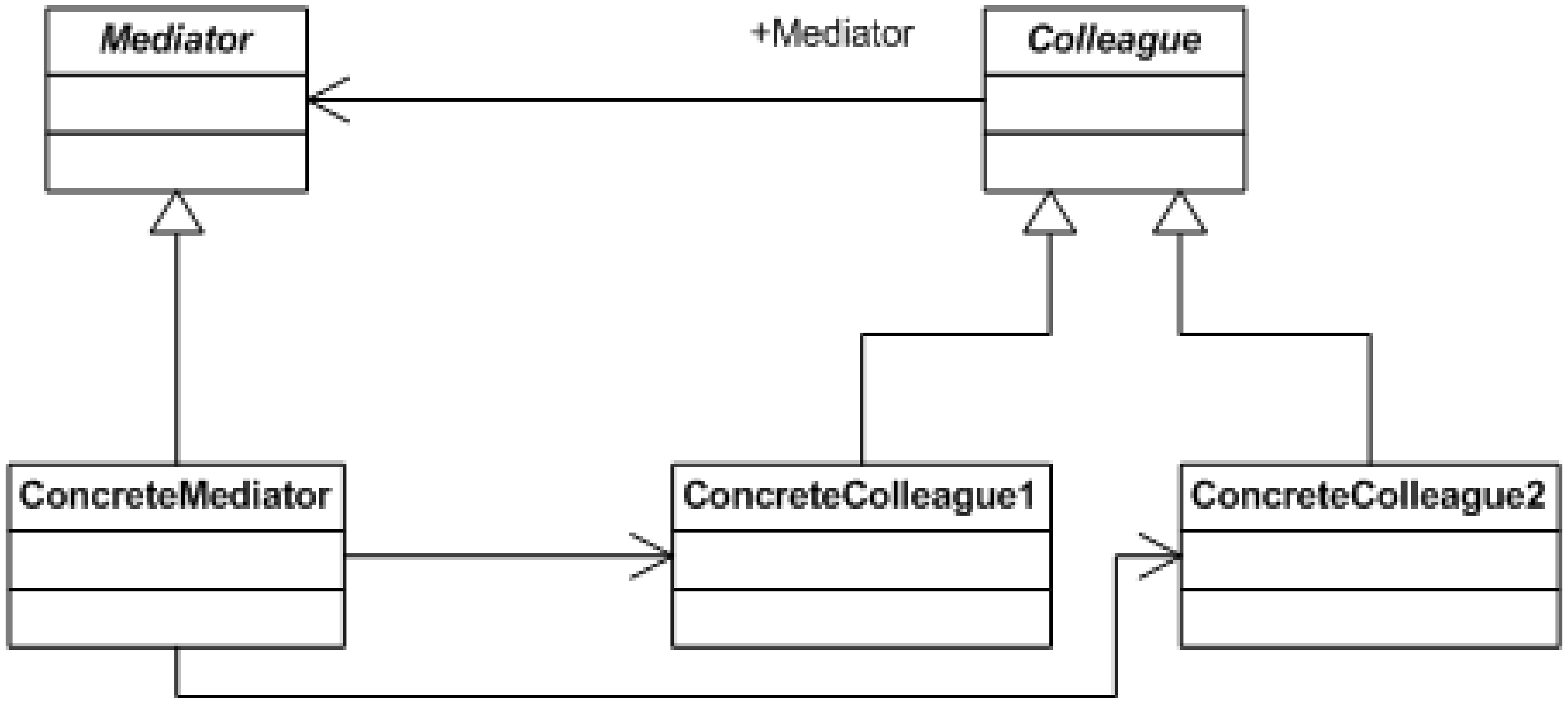
GoF: Mediator

Applicability

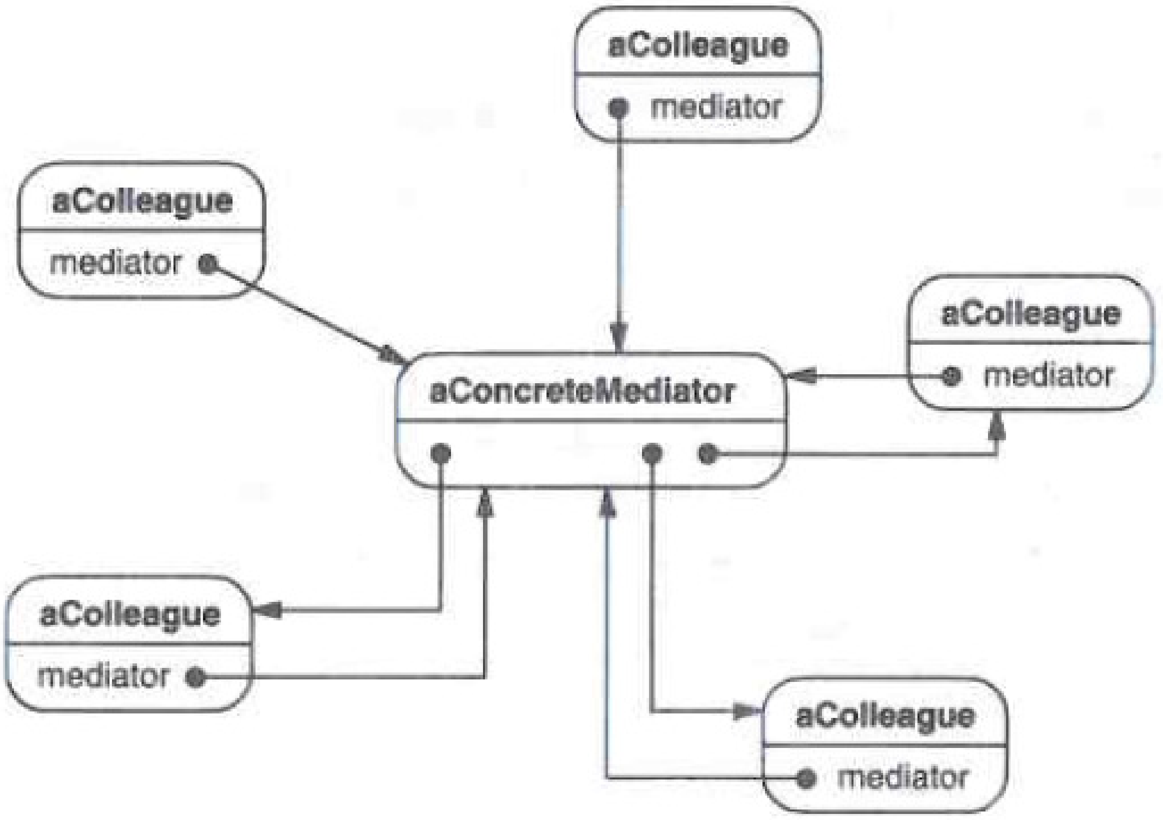
- When a set of objects communicates in a well-defined, but complex way
- When reusing an object is difficult because it refers to and communicates with many other objects (tight coupling)
- When a behavior that is distributed among several classes should be customizable without a lot of subclassing


ATC Mediator
Flight 111
Flight 1011
Flight 112
Flight 747

Mediator: structure



Structure


aColleague
mediator
aColleague
mediator
aColleague
mediator
aConcreteMediator
aColleague
mediator
aColleague
mediator

Mediator

- It encapsulates interconnections between objects
- It is the communications hub
- It is responsible for coordinating and controlling colleague interaction
- It promotes loose coupling between classes
  - By preventing from referring to each other explicitly
- It arbitrates the message traffic

How to use Mediator

1. Identify a collection of interacting objects whose interaction needs simplification
2. Get a new abstract class that encapsulates that interaction
3. Create a instance of that class and redo the interaction with that class alone

Consequences

- Limits subclassing
  - Localizes behavior that would be otherwise distributed among many objects
  - Changes in behavior require changing only the Mediator class
- Decouples colleagues
  - Colleagues become more reusable.
  - You can have multiple types of interactions between colleagues, and you don’t need to subclass or otherwise change the colleague class to do that.

Consequences

- Simplifies object protocols
  - Many-to-many interactions replaced with one-to-many interactions
  - More intuitive
  - More extensible
  - Easier to maintain
- Abstracts object cooperation
  - Mediation becomes an object itself
  - Interaction and individual behaviors are separate concepts that are encapsulated in separate objects

Consequences

- Centralizes control
  - Mediator can become very complex
  - With more complex interactions, extensibility and maintenance may become more difficult
  - Using a mediator may compromise performance

Implementation Issues

- Omitting the abstract Mediator class – possible when only one mediator exists
- Strategies for Colleague-Mediator communication
  - Observer class
    - The colleagues are the subjects: any change in their state is notified to the coordinator that may notify other colleagues.
  - Pointer / other identifier to "self" passed from colleague to mediator, so that the mediator can identify the sender.

Related Patterns

- Façade
  - Unidirectional rather than cooperative interactions between object and subsystem
  - Mediator is like a multi-way Façade pattern.
- Observer
  - May be used as a means of communication between Colleagues and the Mediator

Coordination Languages

- "Mediator" constructs as language primitives:
  - Linda and tuple spaces: late 80's early 90's
    - Middleware acting as a coordinator
- BPEL (Business Process Execution Language) and web services (BPEL4WS o WS-BPEL)

Homework

- This exercise wants to demonstrate the Mediator pattern facilitating loosely coupled communication between different Participants registering with a Chatroom.
  - The Chatroom is the central hub through which all communication takes place.
  - Implement the Chatroom, having the following interface:

    ```
    public interface AbstractChatroom  {
        public abstract void register(Participant participant);
        public abstract void send(String from, String to, String msg);     }
    ```

  - At this point only one-to-one communication is implemented in the Chatroom.
  - Optional: experiment with one-to-many. (communication to a group)