Mediator

BRIAN LAMARCHE

COMPUTER SCIENCE 323 – SOFTWARE DESIGN

Project Question

How implement seek and destroy?

Consider your bottom line:

- Need to compete

Consider Risks of not being able to compete

Biggest Risk

F-stamp

Biggest challenge

Always aim at something that will work

- Complete minimal requirements
- Something is better than nothing!

Then add more functionality

- Increased complexity means increased headache
- Iterate, adding more and more complexity…

Biggest challenge

Always aim at something that will work

- Complete minimal requirements
- Something is better than nothing!

Then add more functionality

- Increased complexity
- Iterations can add more and more complexity…

Biggest challenge

Always assume that you will be extending existing functionality, or adding new functionality

Design consideration:

- Modularity
- Reduction of coupling
- Allow yourself to add functionality easily

Target functionality

Phase 1 Control Logic

- Given a set of targets
- Automatically eliminate targets
- By exercising control Of
  - Missile Launcher
  - Target Locations

Target functionality

Phase 2 Control Logic

- Given:
  - A set of targets
  - An image of target area
- Automatically eliminate targets
- By exercising control Of
  - Missile Launcher
  - Target Locations

Mediator

Behavioral

Captures how a set of object interact

Consider that decoupling objects promotes modularity and flexibility

- But who promotes the interaction logic?
- How does communication occur between objects?

Mediator

Modularized objects do not communicate with each other

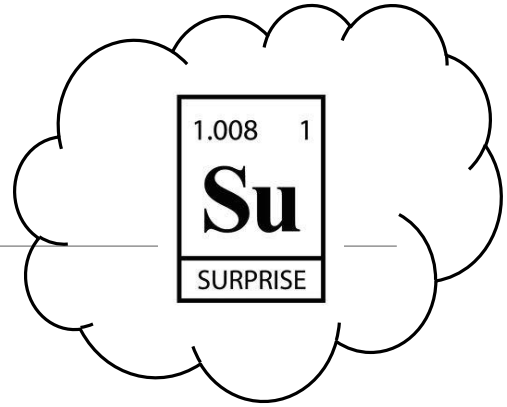
Communicate through a mediator

Defense Objects

Missile Defense

Grumpy Sentry

Defense Objects



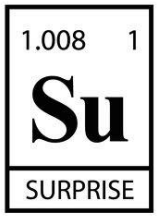
I've got a target in sight, but I don't care

Defense Objects

How do you make the two things talk to each other?

Speaks in sarcasm…

Speaks in elements of



Defense Objects

Don’t couple them…

Defense Objects

How do you make the two things talk to each other?

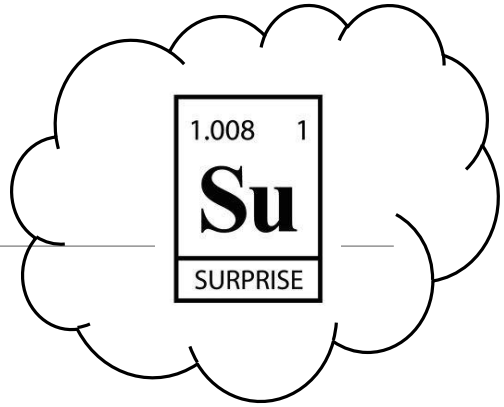
Need some kind of logic…

1. You can fire until you know you have a target
2. Don't give the target defense system more than it needs
   Especially when CN

Defense Objects

Defense Objects

Defense Objects



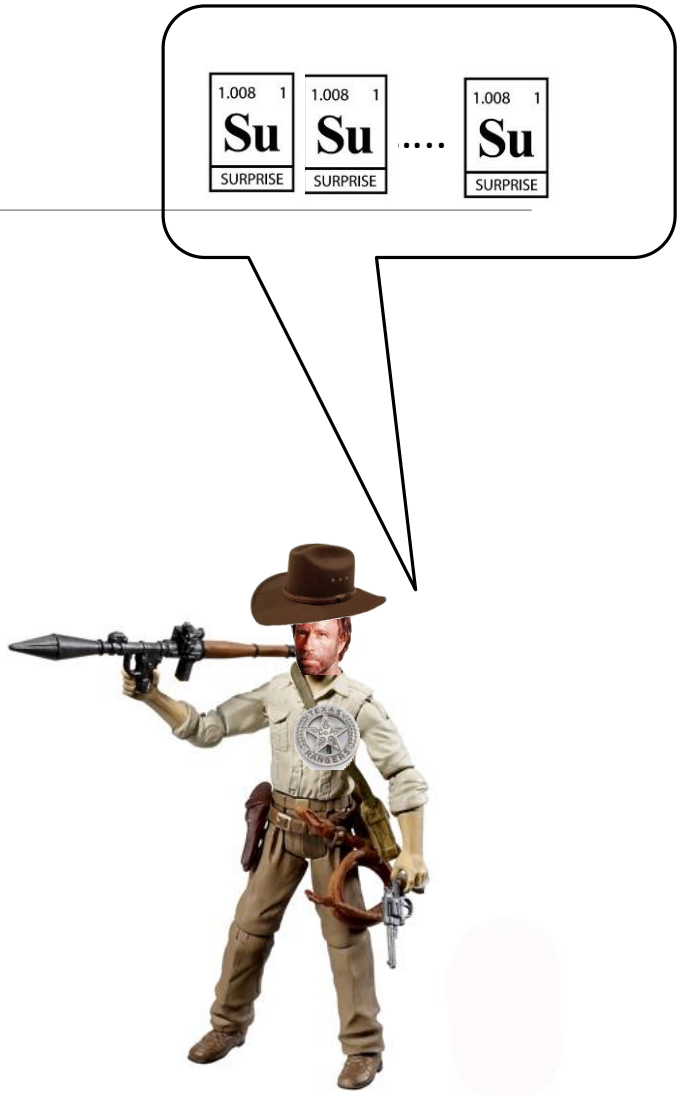
Defense Objects

Defense Objects

TARGET AT 50, 52

1.008 1
Su
SURPRISE

Defense Objects



Defense Objects

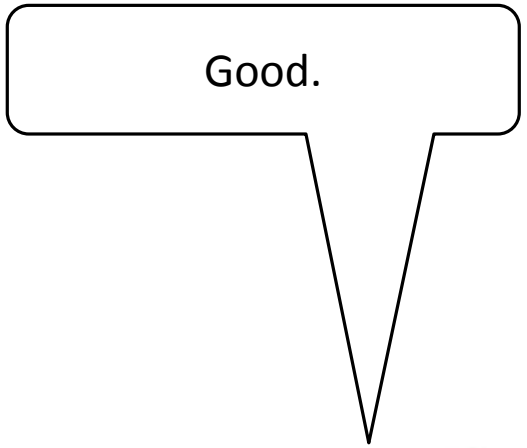
Defense Objects

Defense Objects

Defense Objects

TARGET was destroyed

Defense Objects



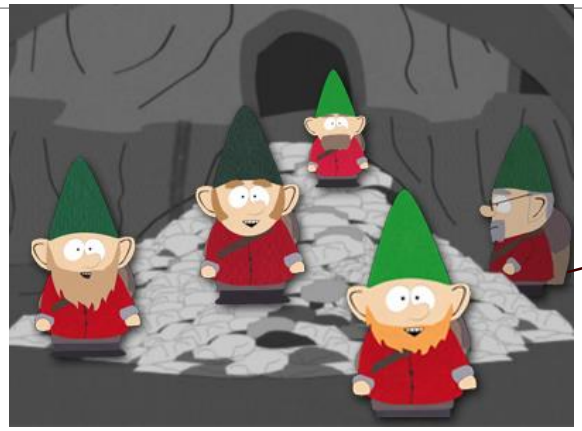
Mediator Pattern

Use in conjunction with

- Observer
- *Strategy – coming soon.*

Using with Observer allows you to change the communication logic between objects

Another Mediator



1. Detect Missiles
2. ?
3. Profit

Players

Mediator

- Concrete Mediator
- Communication and Control logic

Colleague

- Concrete Colleague
- Performs specific tasks, communicates to other systems via mediator

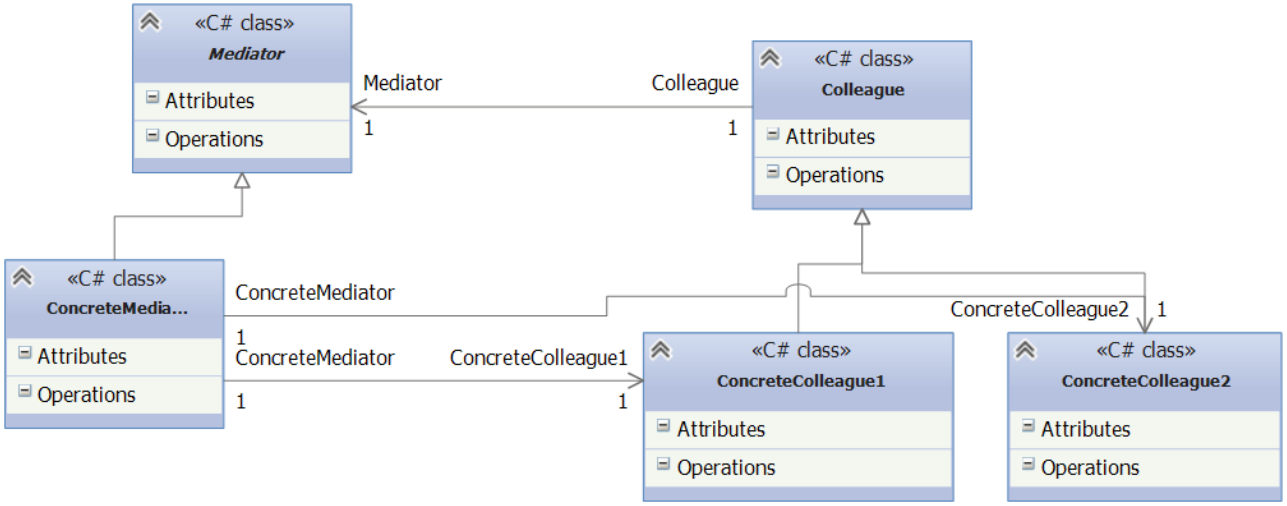
Players

Mediator
- Concrete Mediator

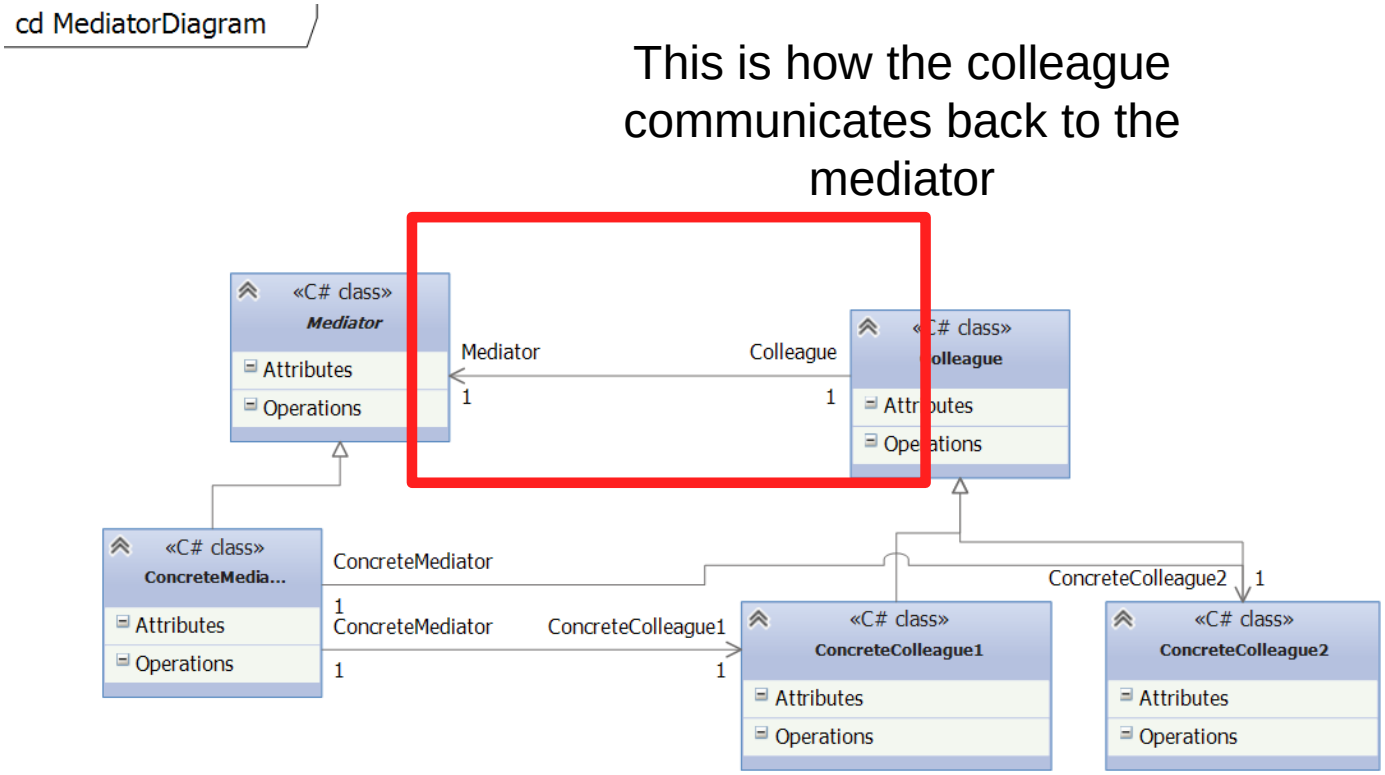
Colleague
- Concrete Colleague

UML

cd MediatorDiagram



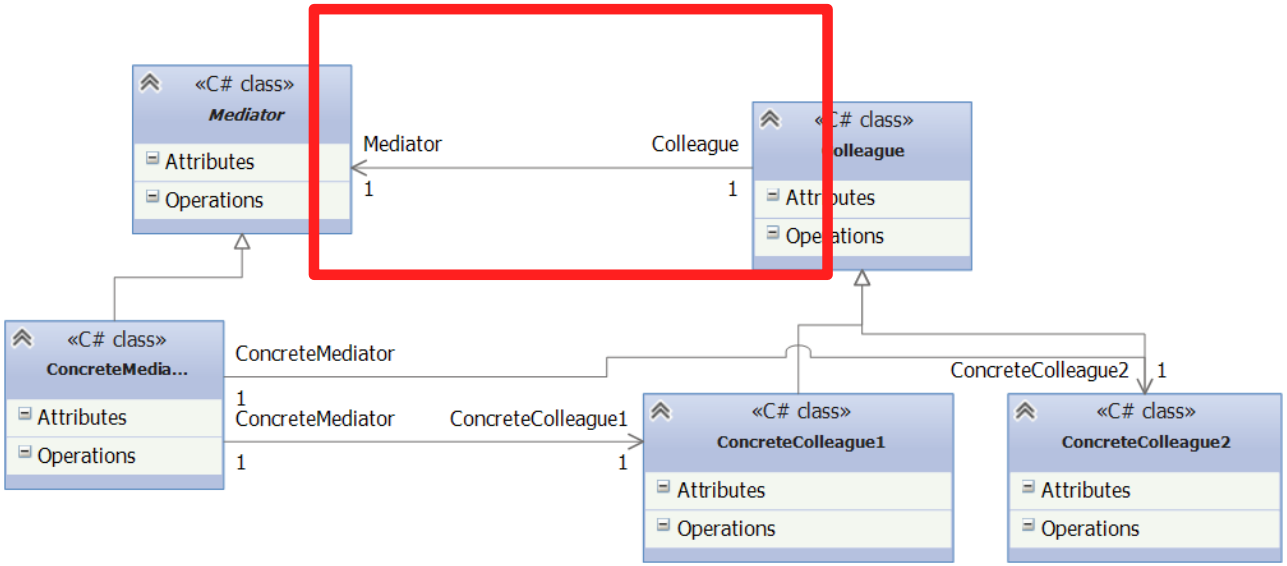
UML


cd MediatorDiagram
This is how the colleague communicates back to the mediator
«C# class»
Mediator
Attributes
Operations
Mediator
1
Colleague
1
«C# class»
Attributes
Operations
«C# class»
ConcreteMedia...
Attributes
Operations
ConcreteMediator
1
ConcreteMediator
1
ConcreteColleague1
1
ConcreteColleague2
1
«C# class»
ConcreteColleague1
Attributes
Operations
«C# class»
ConcreteColleague2
Attributes
Operations

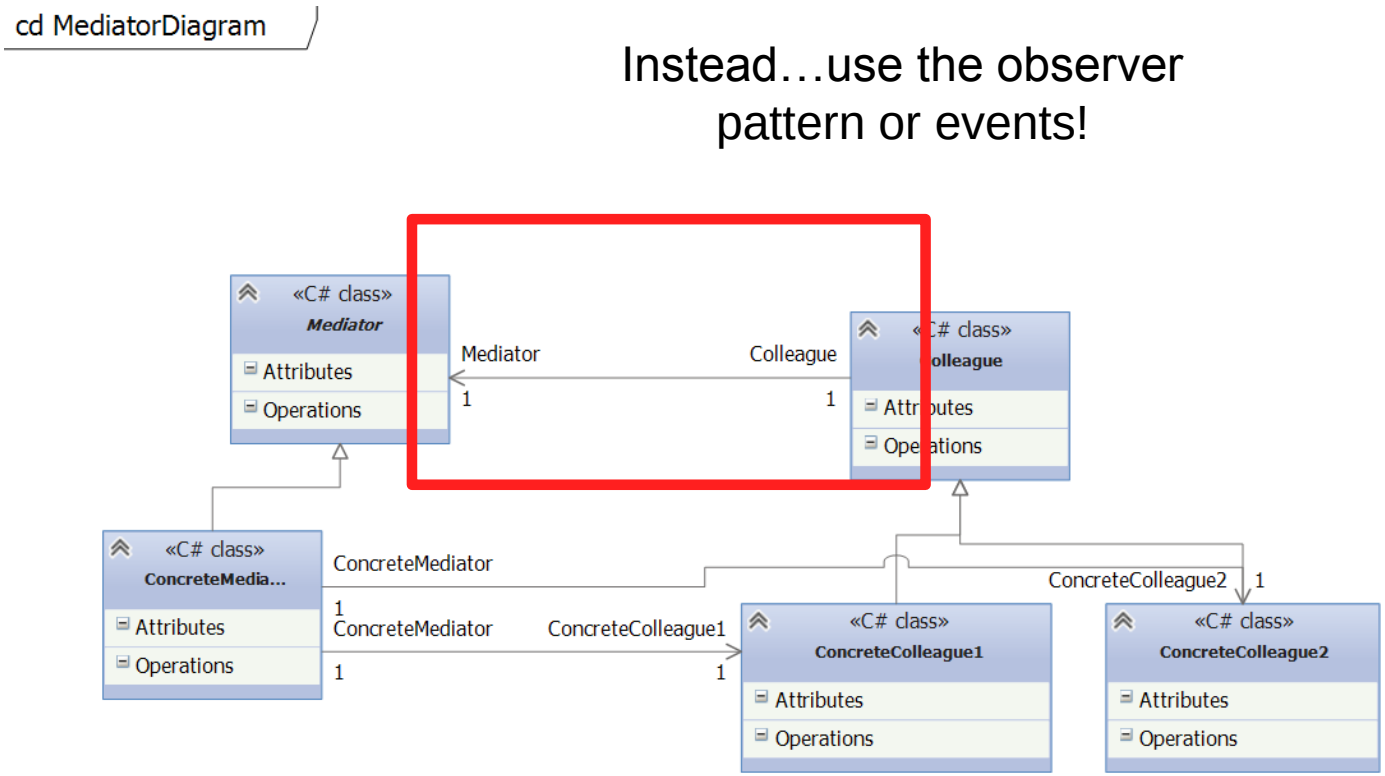
In reality…this structure sucks

cd MediatorDiagram

You've now coupled a colleague to the mediator



In reality…this structure sucks



Examples

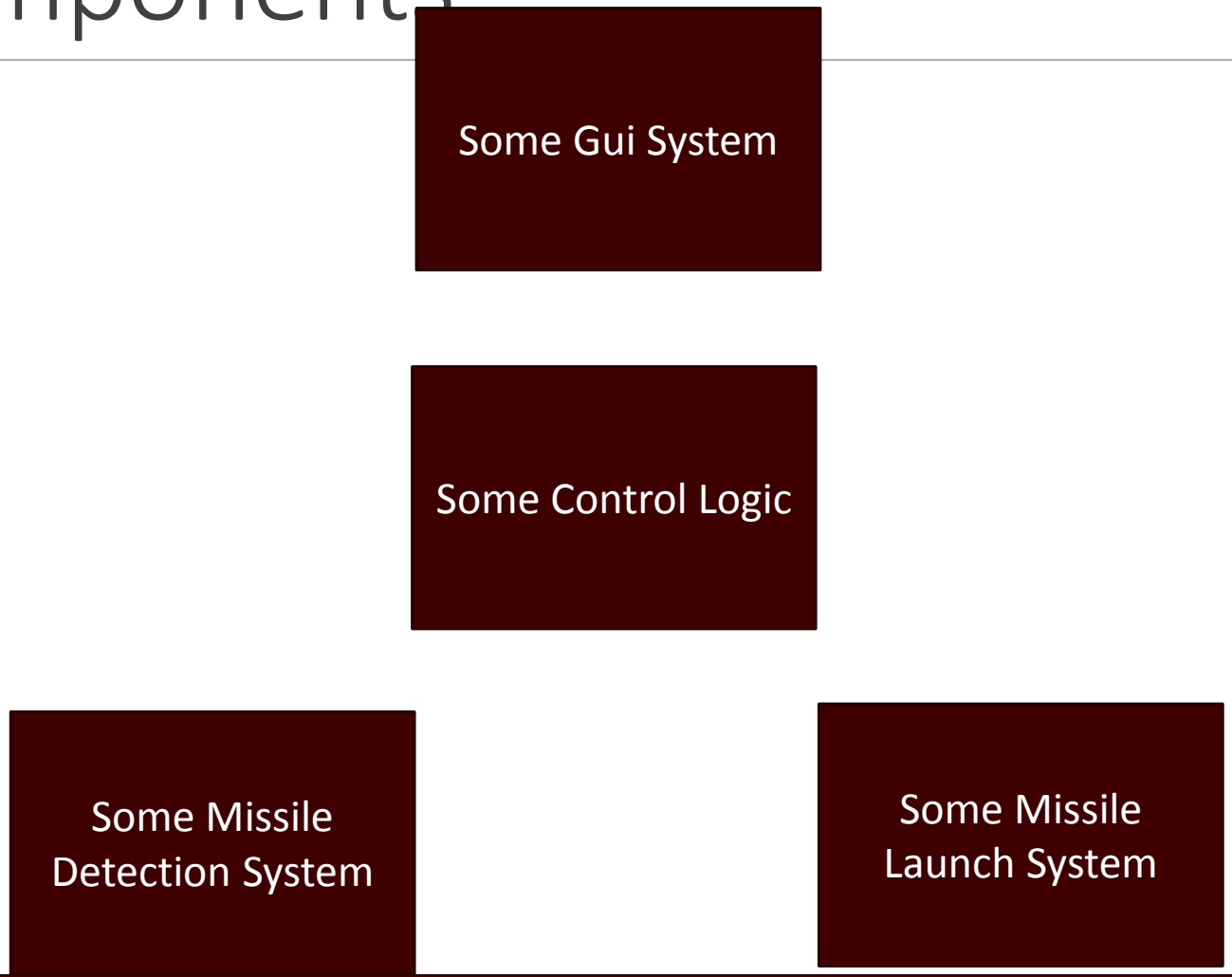
Mediator

In all seriousness, the ability to decouple objects is key.

It allows us to change / swap logic between objects. (Vary interaction independently)

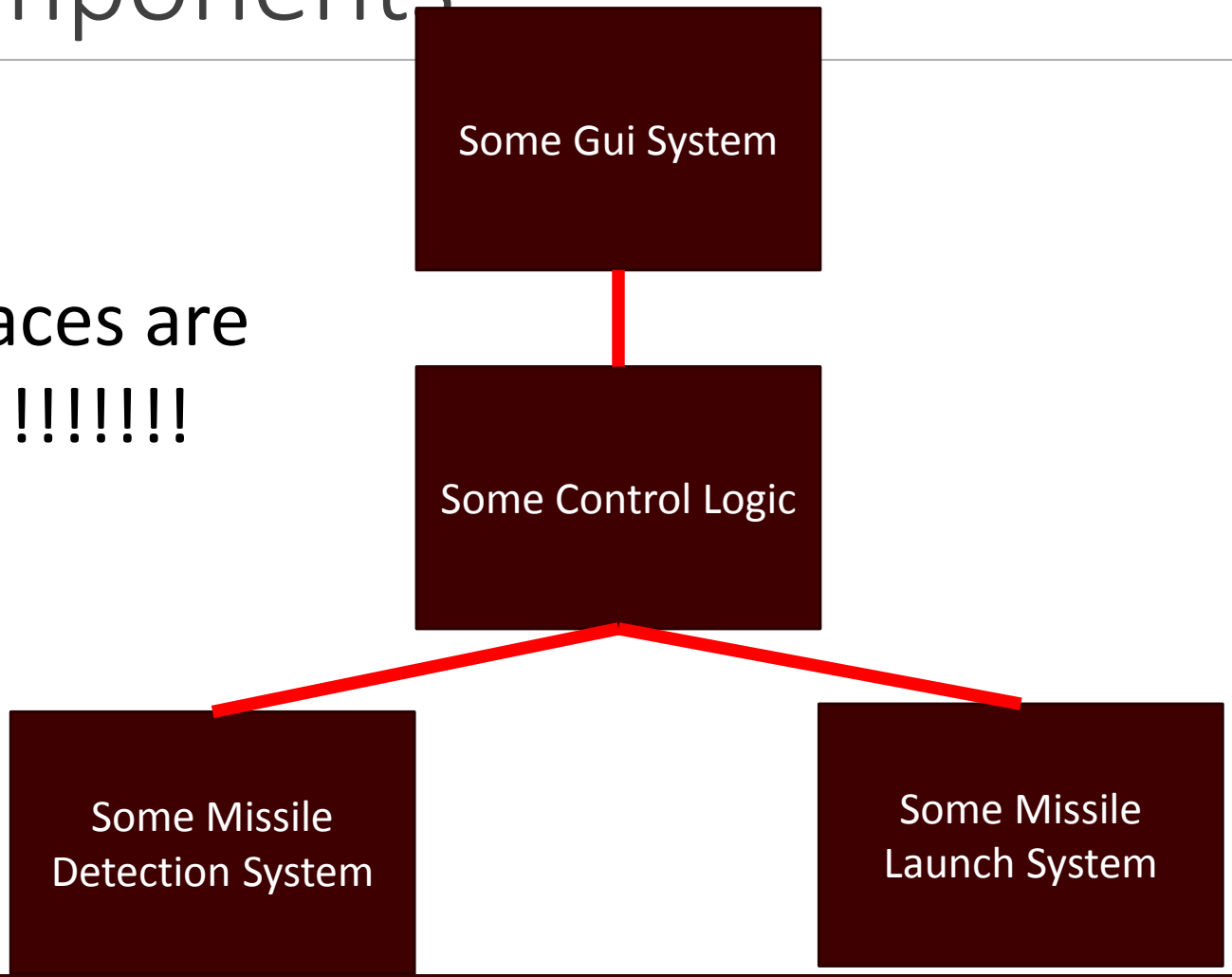
Interfaces / Abstraction allows us to easily perform these tasks.

Components


Some Gui System
Some Control Logic
Some Missile Detection System
Some Missile Launch System

Components

Interfaces are KEY!!!!!!!!



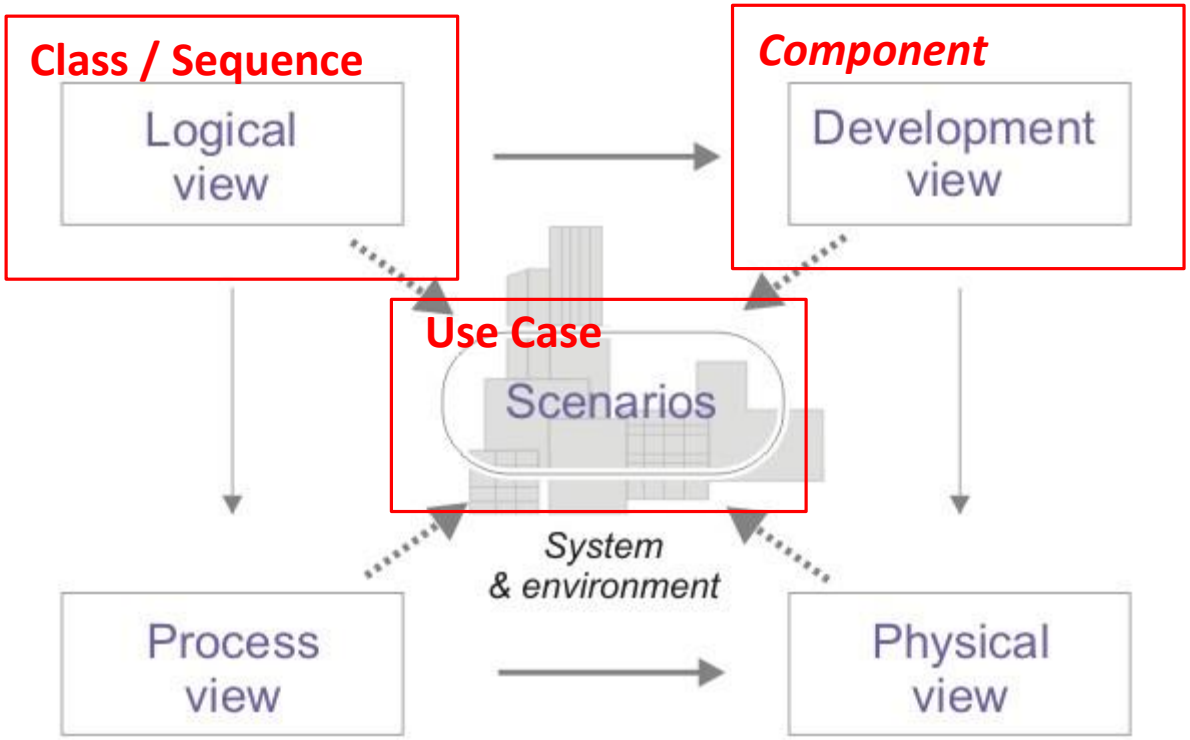


Figure courtesy of Marcel Douwe Dekker – CCAS License 3.0

Design

Add a mediator to your project

- Controls logic between GUI and Rest of System
  - Loading Targets
  - Firing Missiles
- Abstract the mediator first
  - Abstract class or Interface

Model the system first in UML

- You probably already have most of this done

Use Case

Actor

- Instructor

Pre-conditions

- Application started
- Application in IDLE State

Use Case

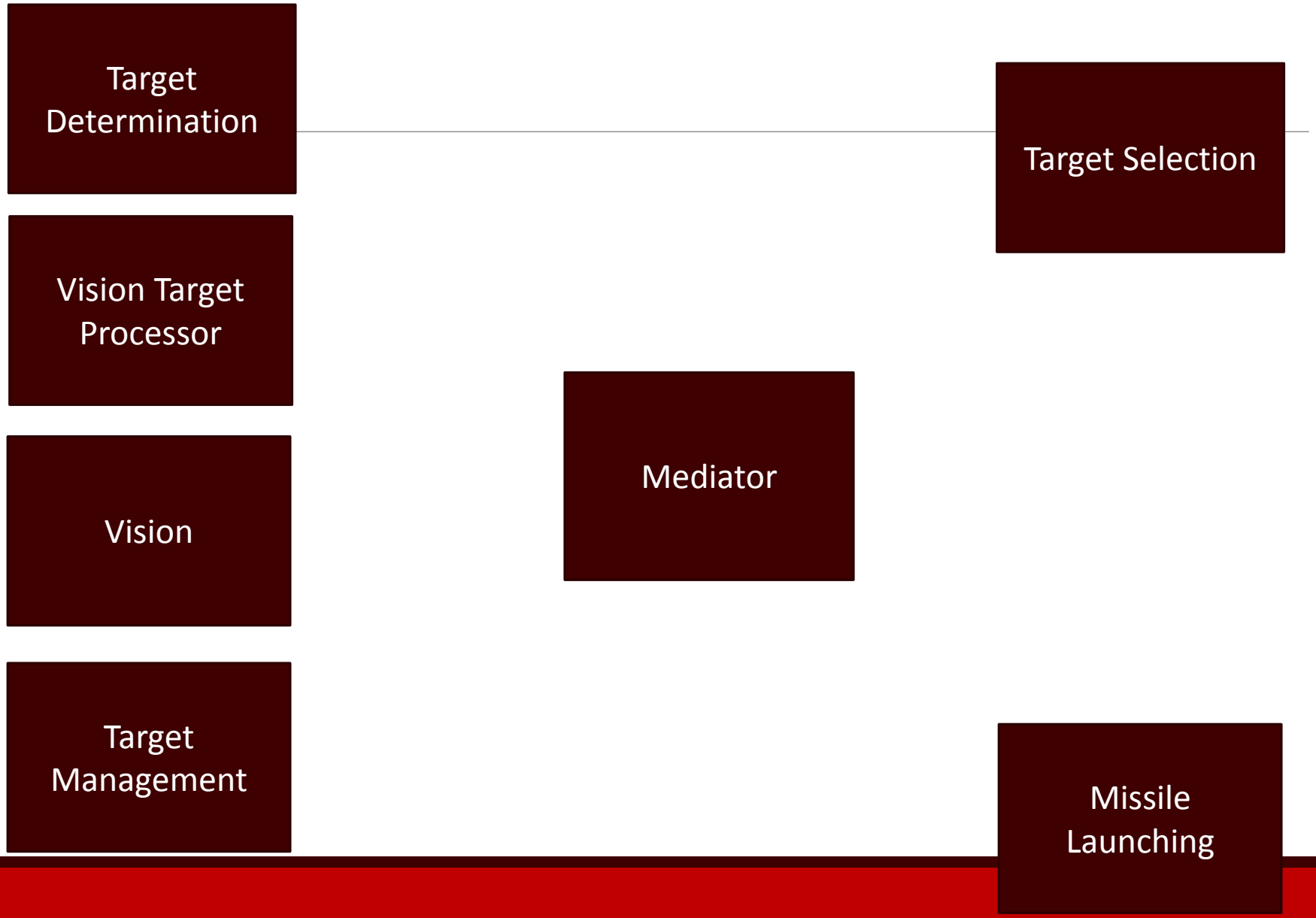
Actions:

- User clicks on the "*Load Targets"* button that displays an open file dialog window.
- User selects a file from their hard drive
- The program reads the target file and loads it into the GUI
- The user then presses the "Fire At Missiles" button
- The program starts to shoot at targets loaded by the program
- The program displays how many missiles are left to be fired
- The program displays how many targets destroyed

Use Case

Post Conditions:

- The targets have been eliminated.
- The program resides to an IDLE state.
- The program displays the total run time on the screen.


Target Determination
Target Selection
Vision Target Processor
Mediator
Vision
Target Management
Missile Launching

Next

Next Tuesday

- UML Component Diagrams