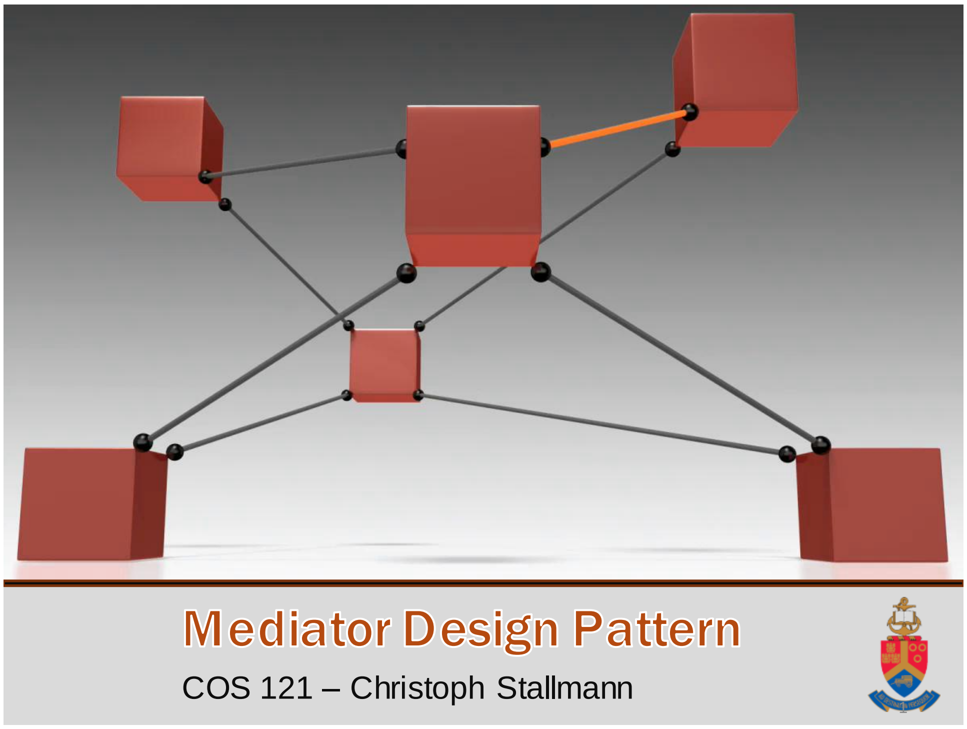
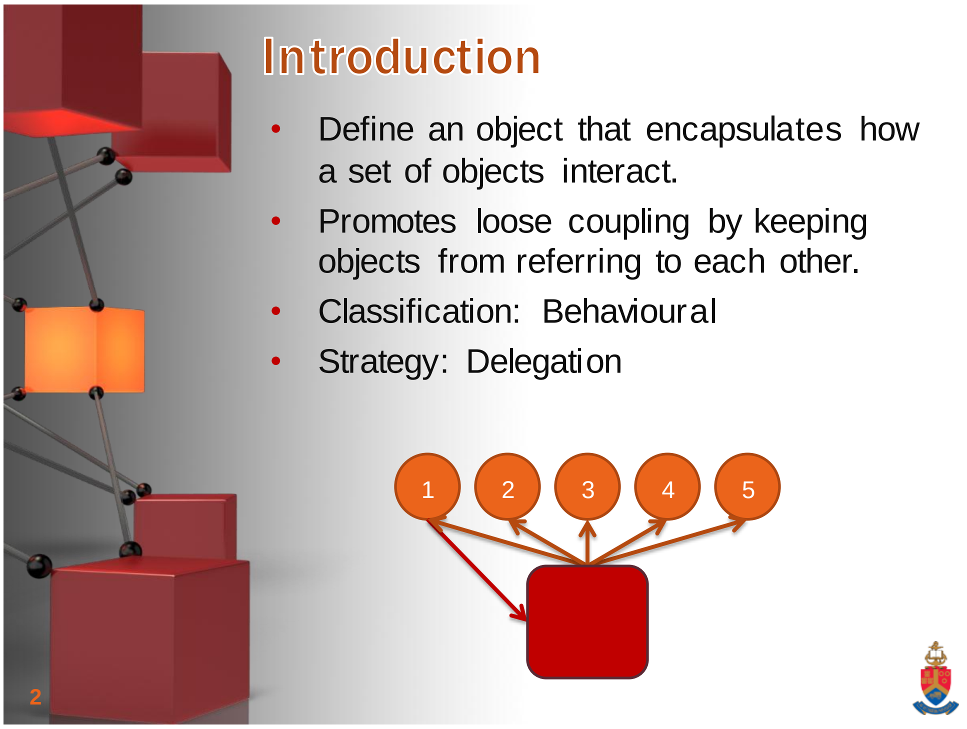


Mediator Design Pattern

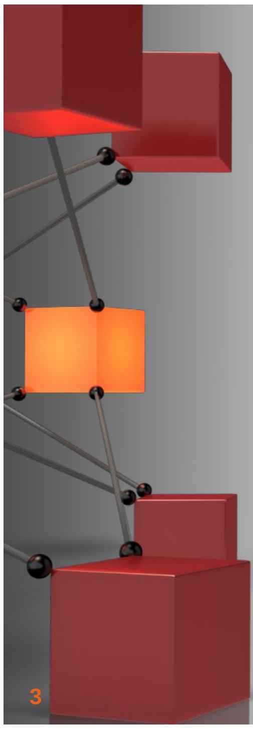
COS 121 – Christoph Stallmann



Introduction

- Define an object that encapsulates how a set of objects interact.
- Promotes loose coupling by keeping objects from referring to each other.
- Classification: Behavioural
- Strategy: Delegation

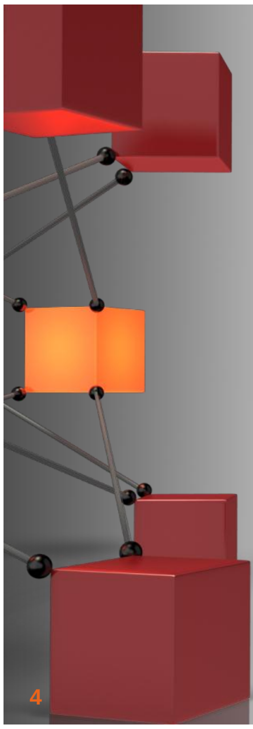
2



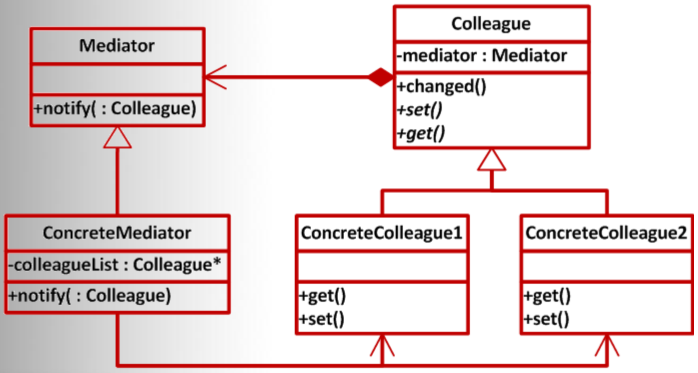
Reason

- Eliminate dependencies between potential reusable pieces.
  - Referred to as spaghetti phenomena.
- Inform all other objects if one object changes.
- Replace a many-to-many relationship with:
  - A number of one-to-one relationships.
  - One one-to-many relationship.

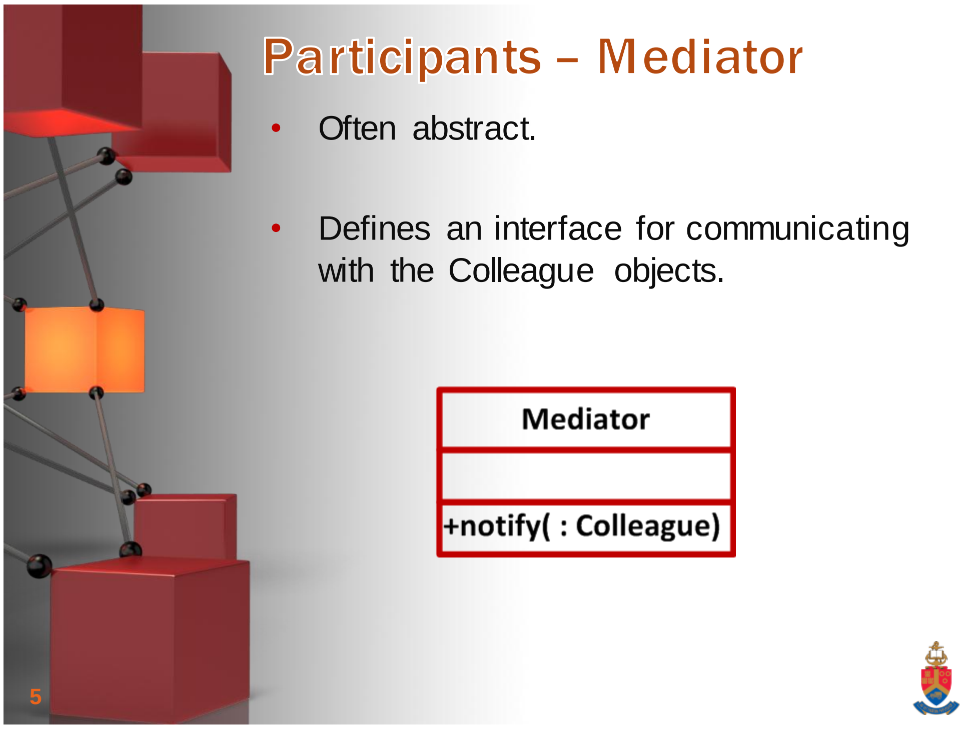
3



Structure



4

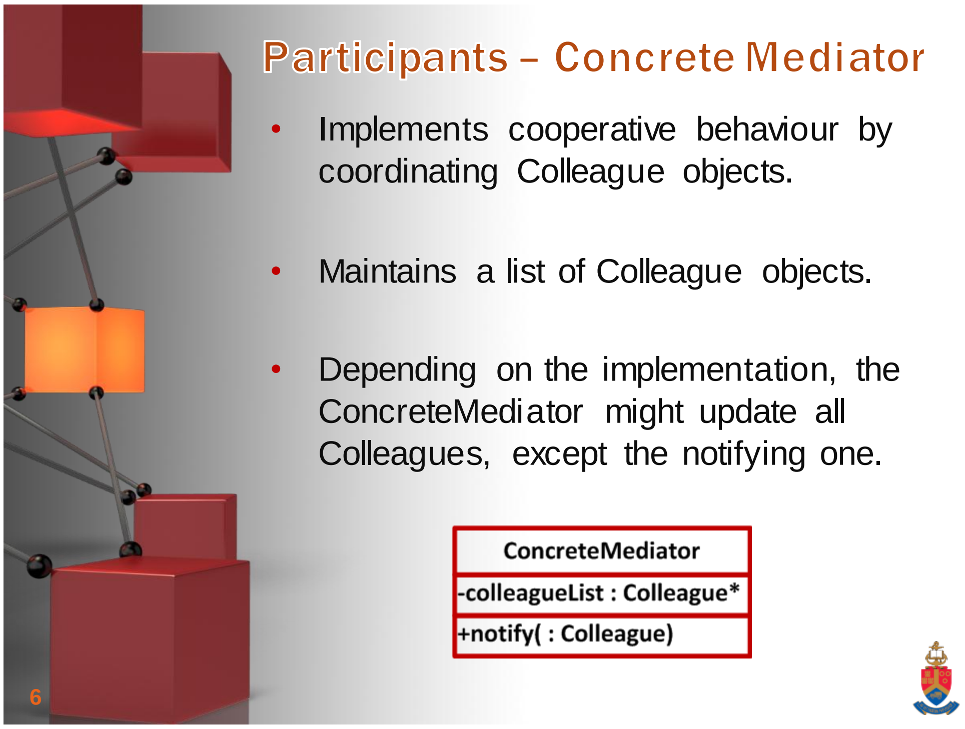


Participants – Mediator

- Often abstract.
- Defines an interface for communicating with the Colleague objects.

Mediator
+notify(: Colleague)

5

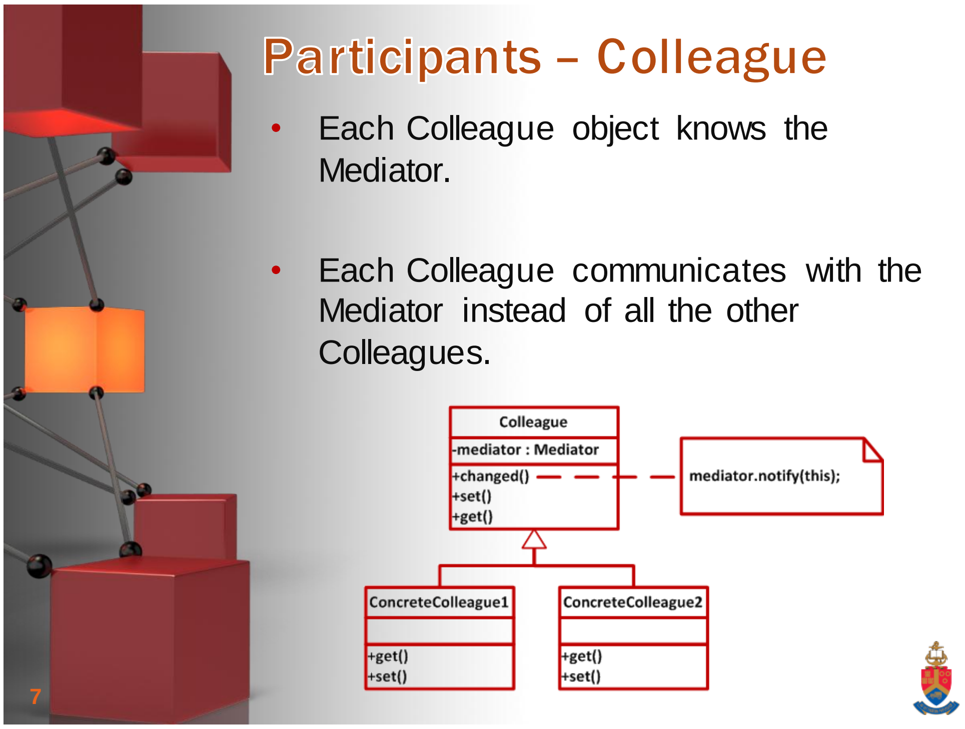


Participants – Concrete Mediator

- Implements cooperative behaviour by coordinating Colleague objects.
- Maintains a list of Colleague objects.
- Depending on the implementation, the ConcreteMediator might update all Colleagues, except the notifying one.

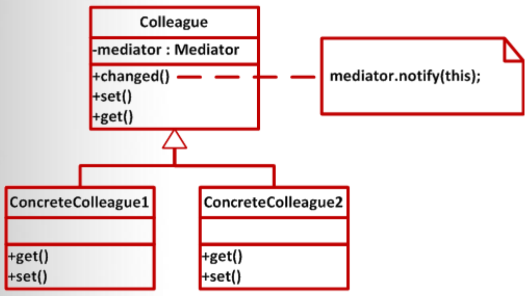
ConcreteMediator
-colleagueList : Colleague*
+notify(: Colleague)

6

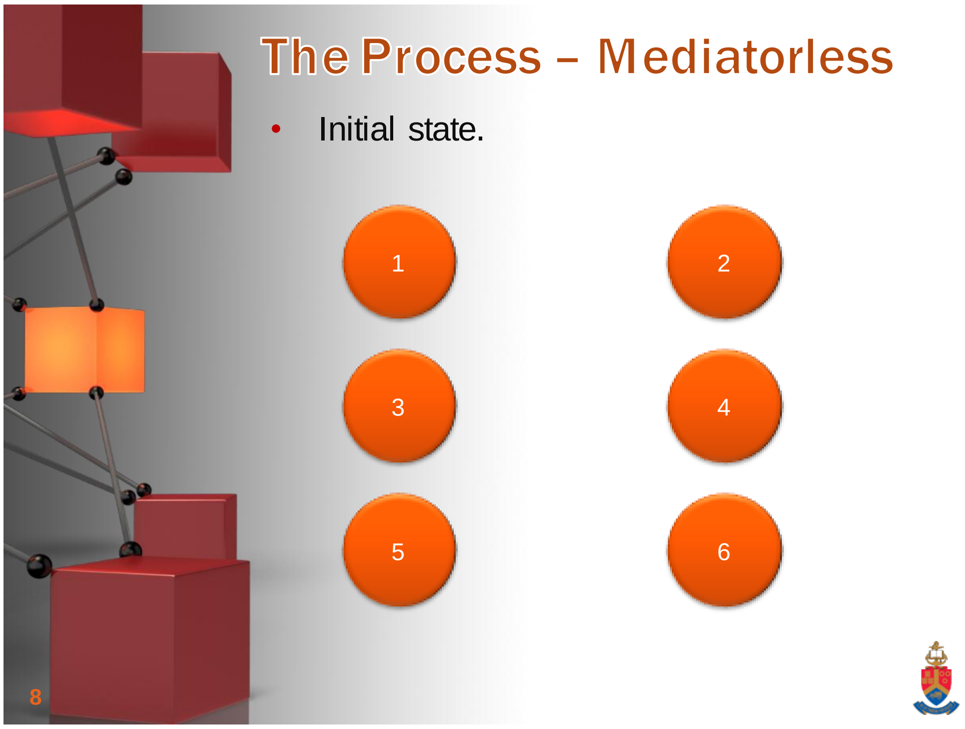


Participants – Colleague

- Each Colleague object knows the Mediator.
- Each Colleague communicates with the Mediator instead of all the other Colleagues.

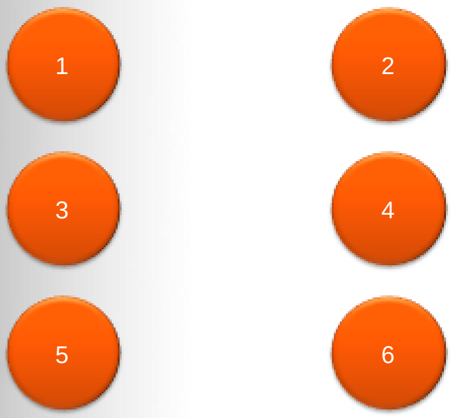


7



The Process – Mediatorless

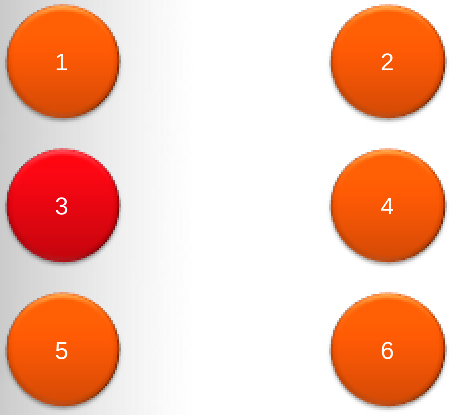
- Initial state.



8

The Process – Mediatorless

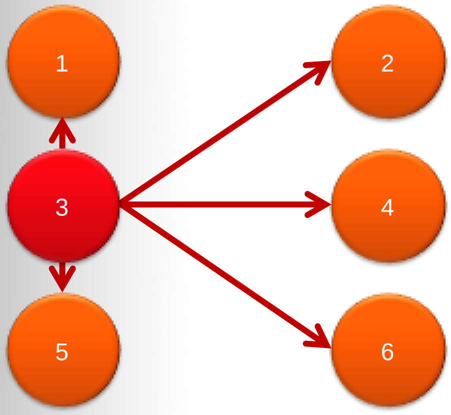
- One of the object changes.



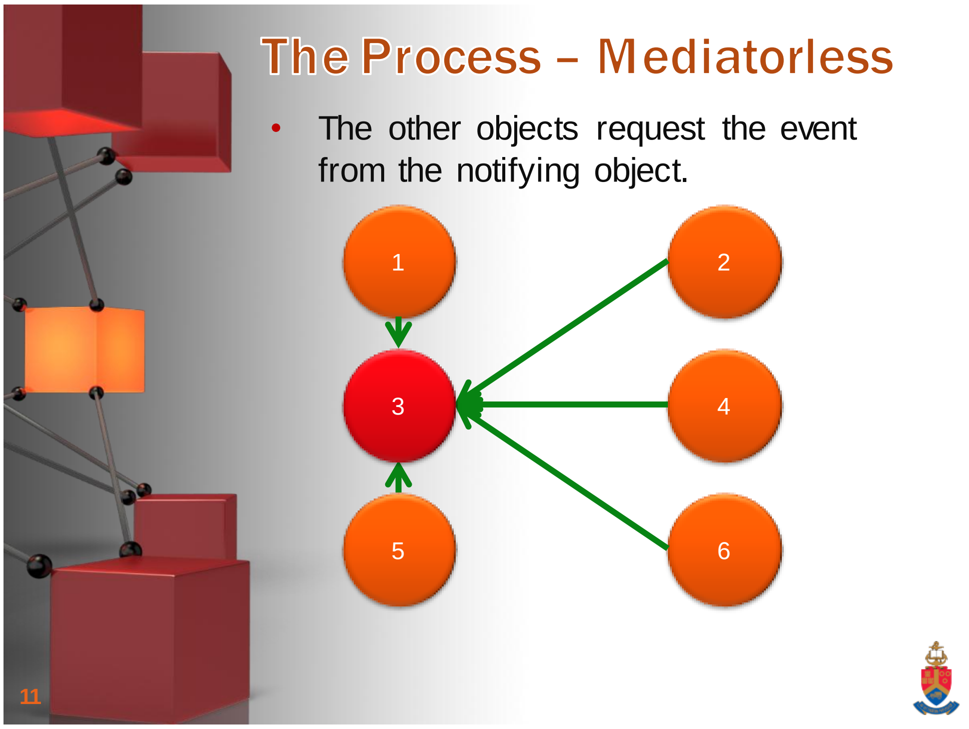
9

The Process – Mediatorless

- The object notifies all other objects.

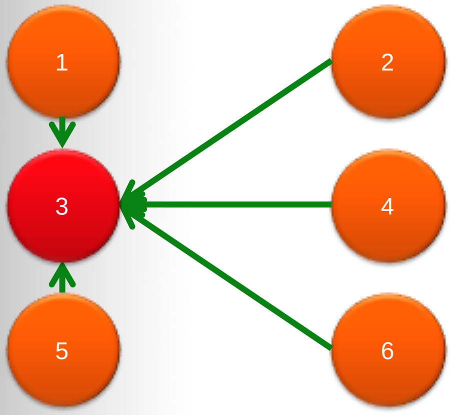


10

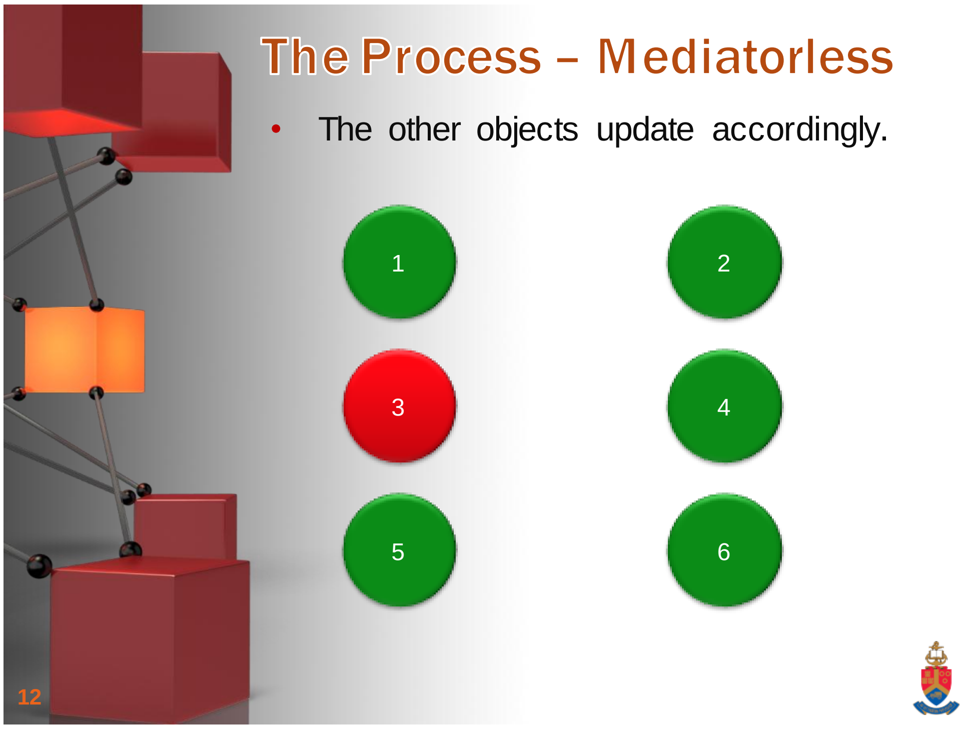


The Process – Mediatorless

- The other objects request the event from the notifying object.

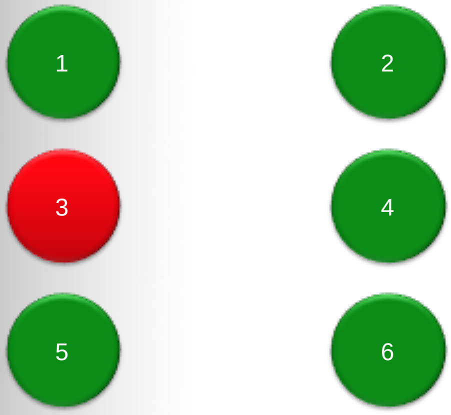


11

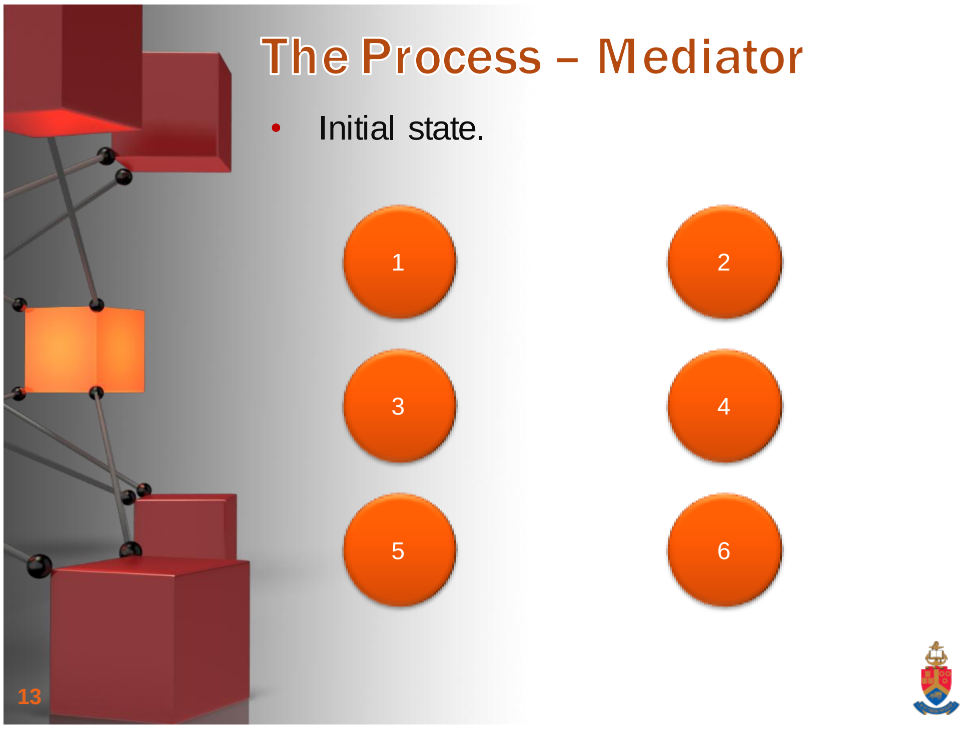


The Process – Mediatorless

- The other objects update accordingly.

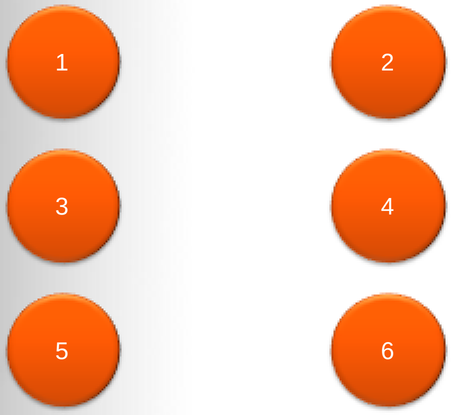


12

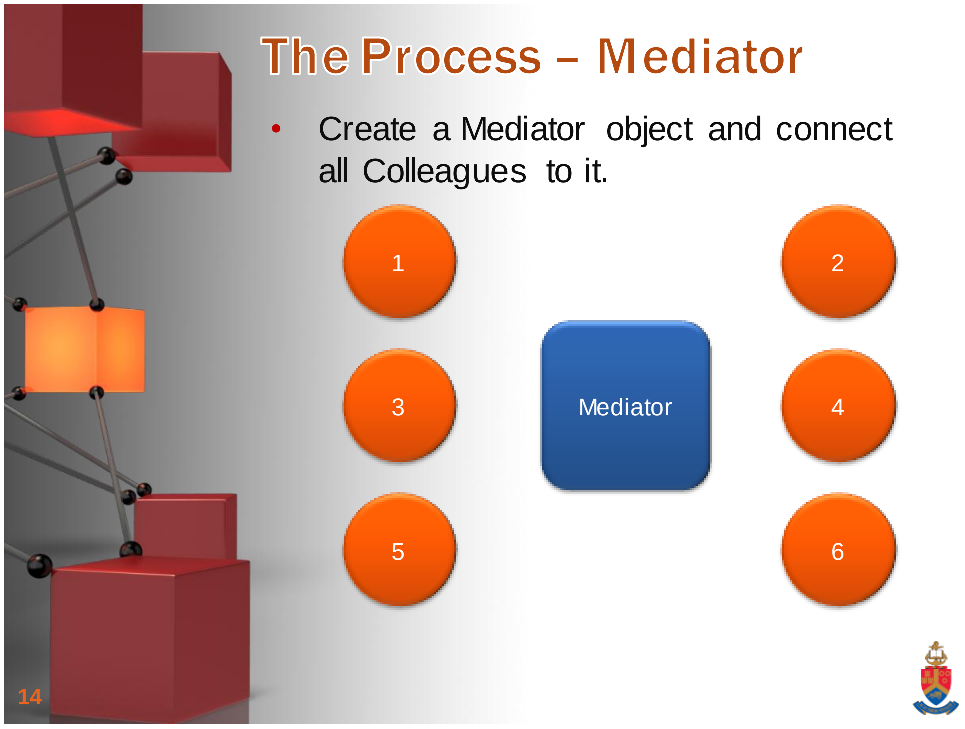


The Process – Mediator

- Initial state.

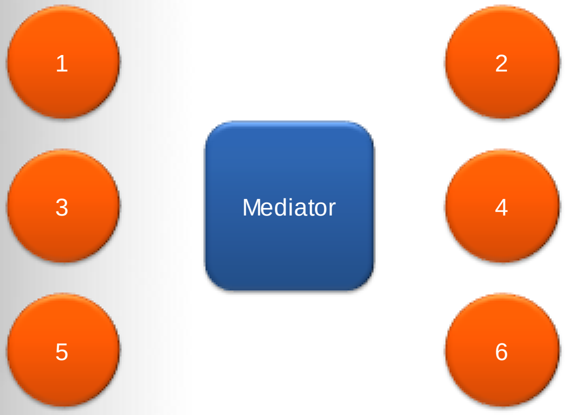


13

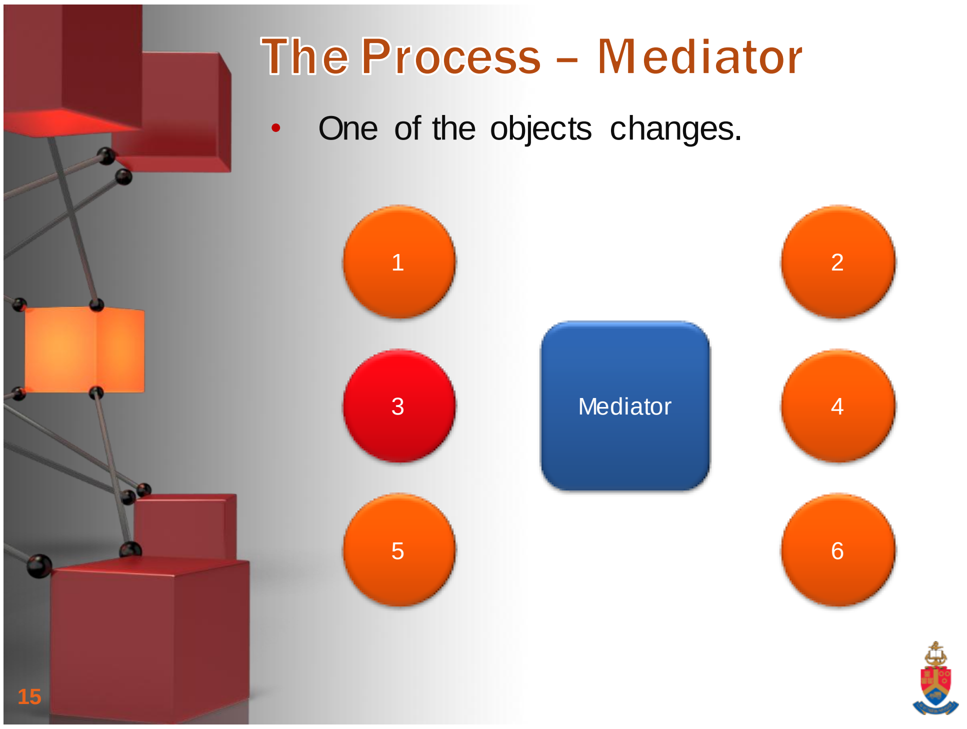


The Process – Mediator

- Create a Mediator object and connect all Colleagues to it.

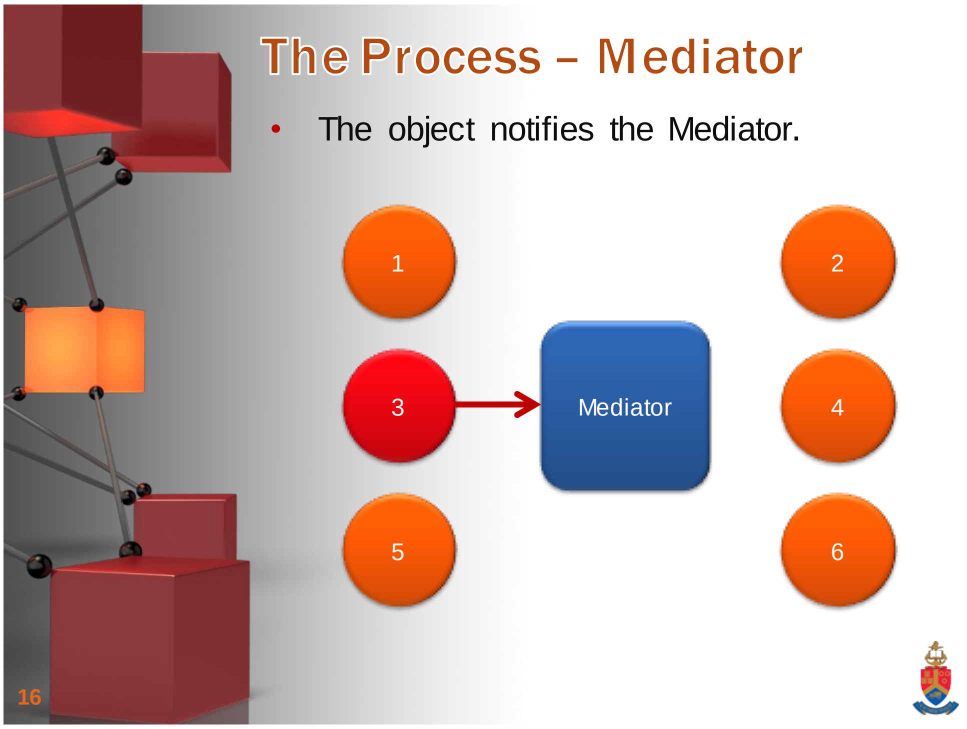


14



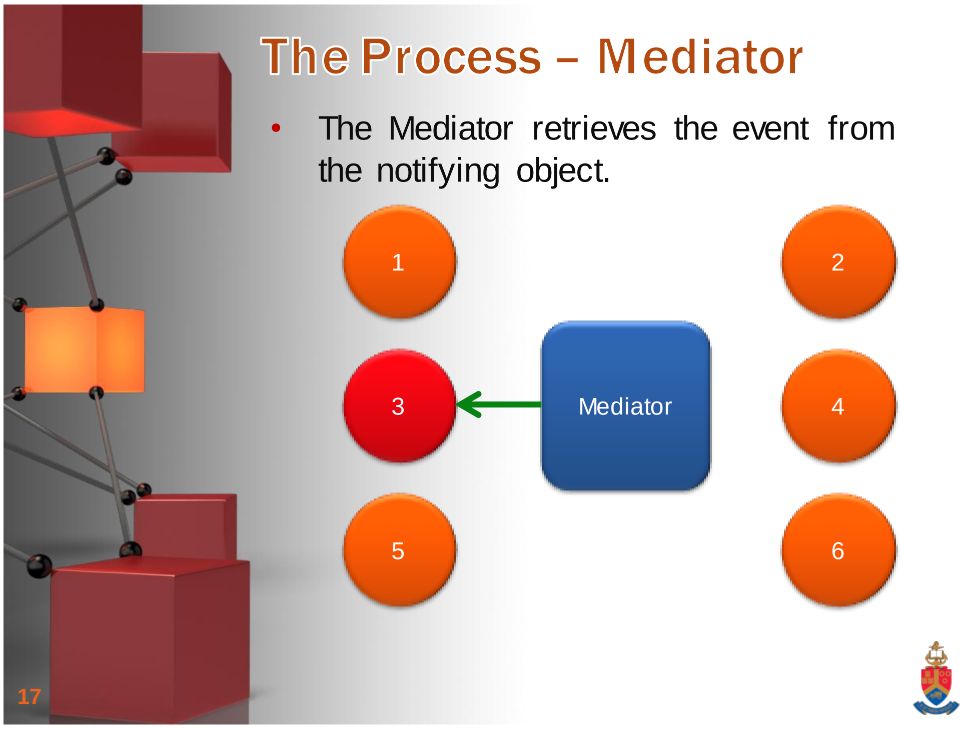
The Process – Mediator

- One of the objects changes.



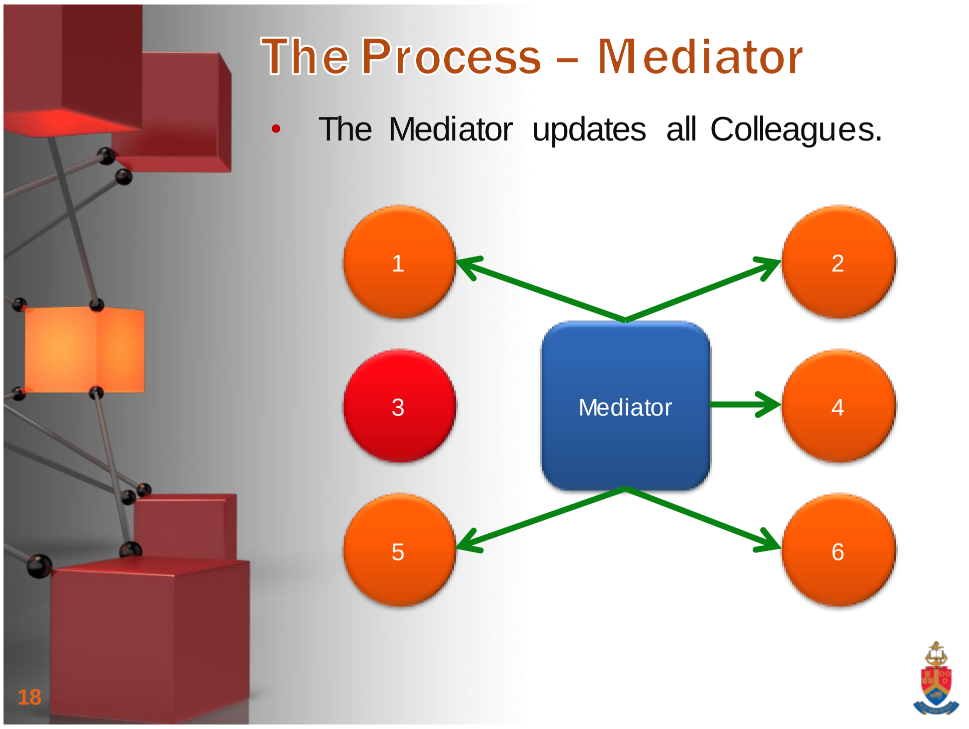
The Process – Mediator

- The object notifies the Mediator.



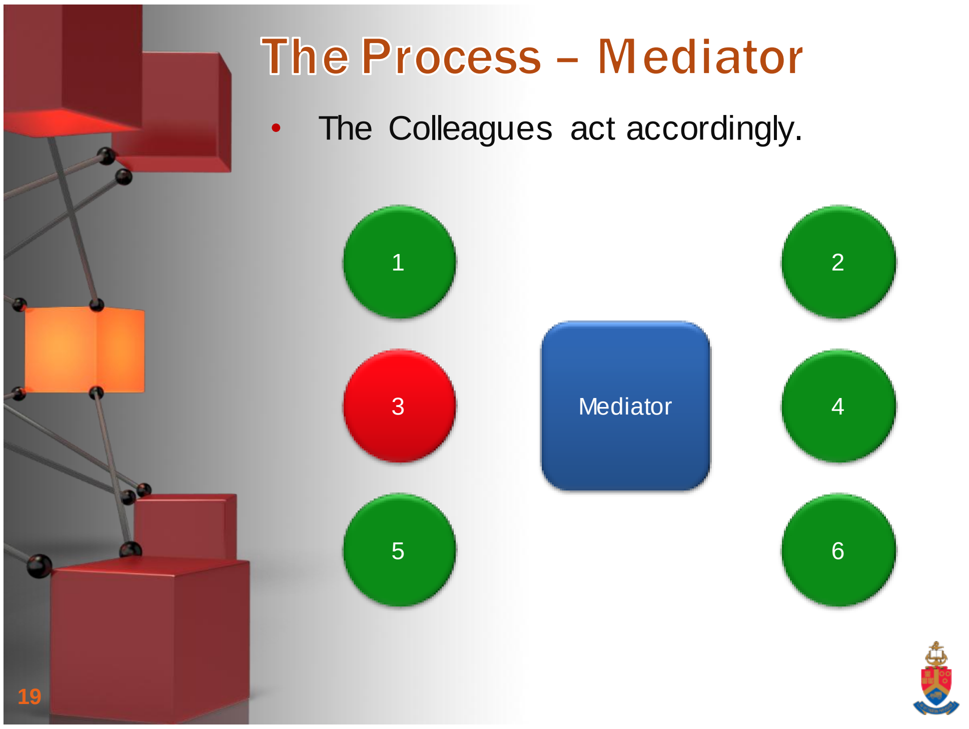
The Process – Mediator

- The Mediator retrieves the event from the notifying object.



The Process – Mediator

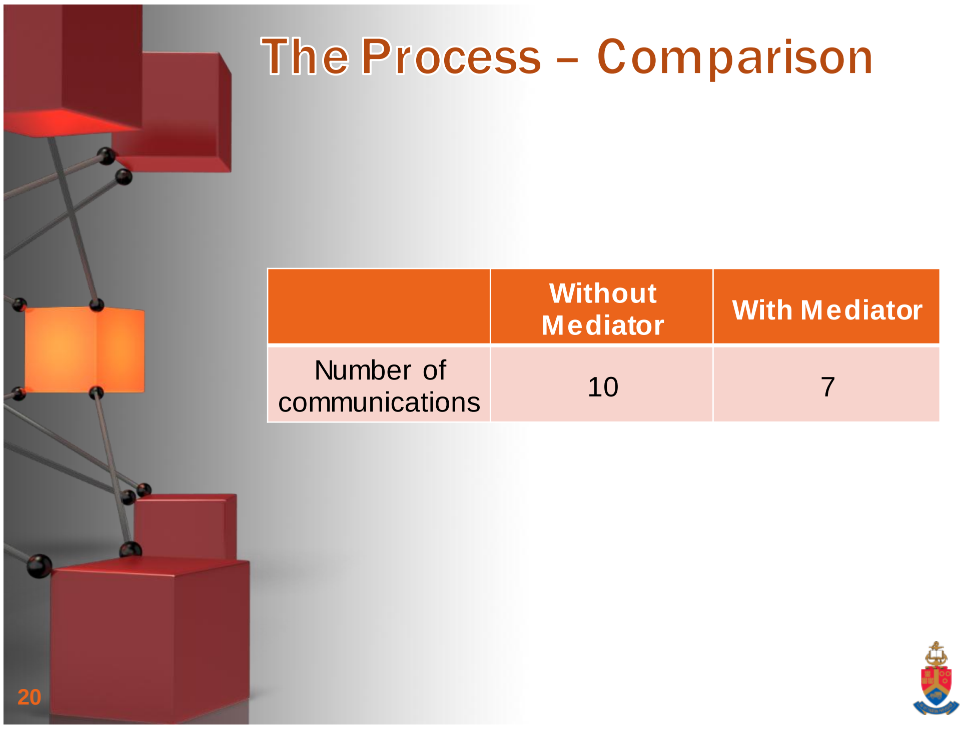
- The Mediator updates all Colleagues.



The Process – Mediator

- The Colleagues act accordingly.

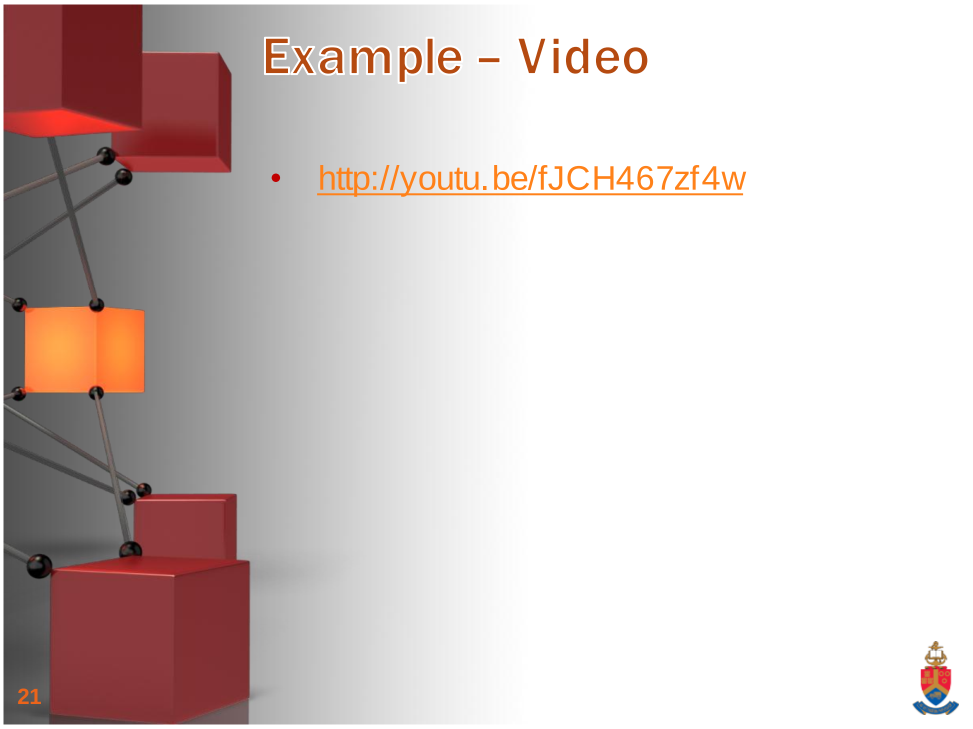
19



The Process – Comparison

	Without Mediator	With Mediator
Number of communications	10	7

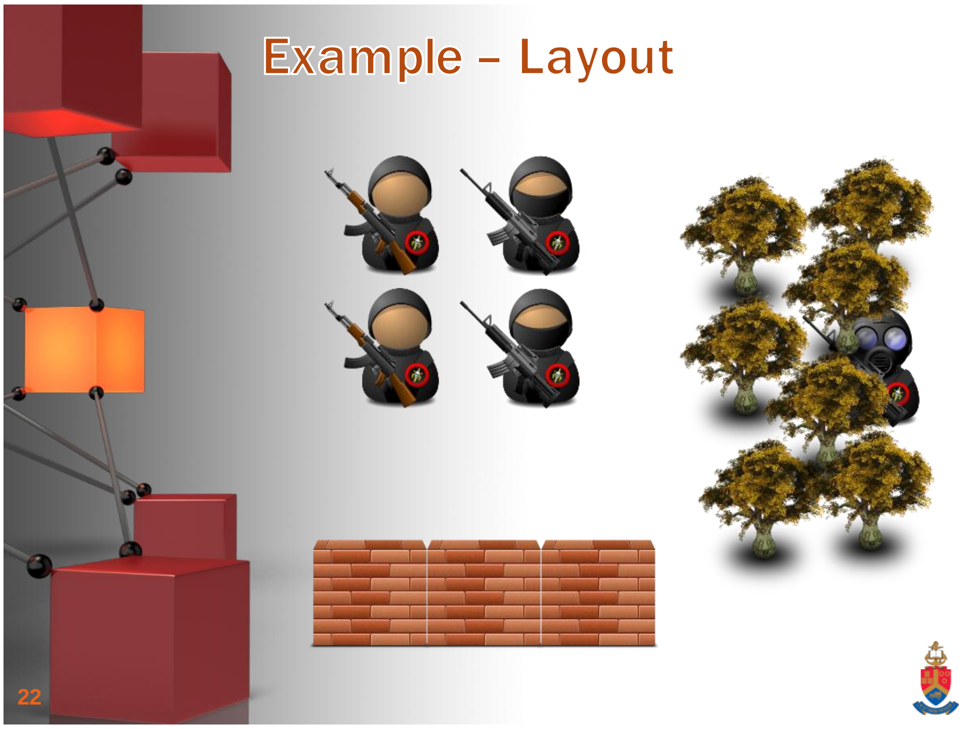
20



Example – Video

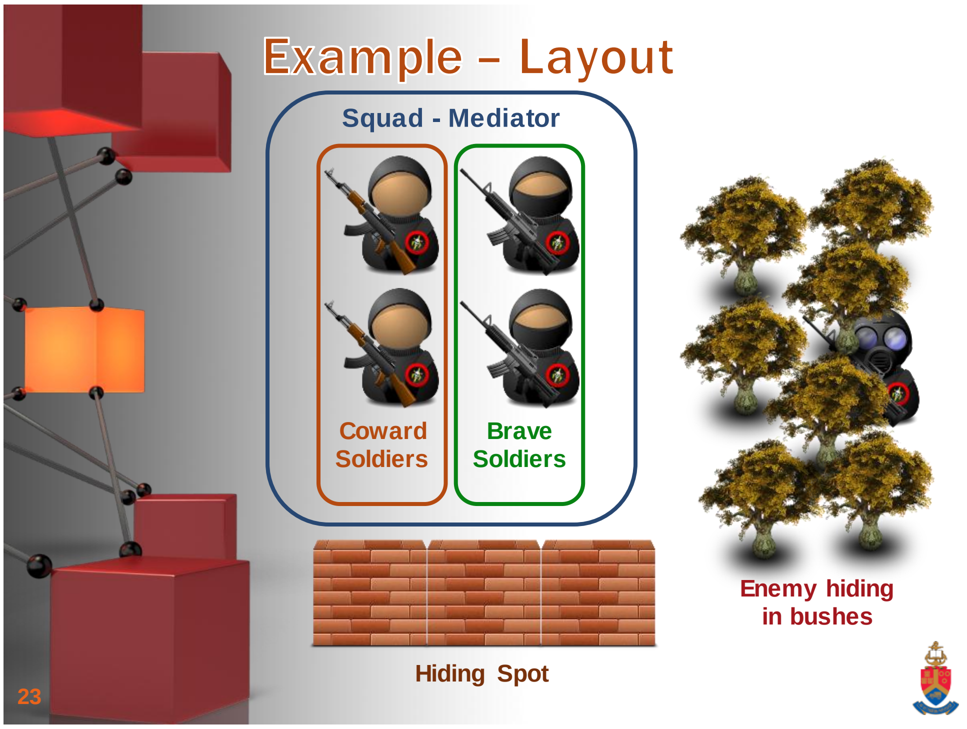
- http://youtu.be/fJCH467zf4w

21

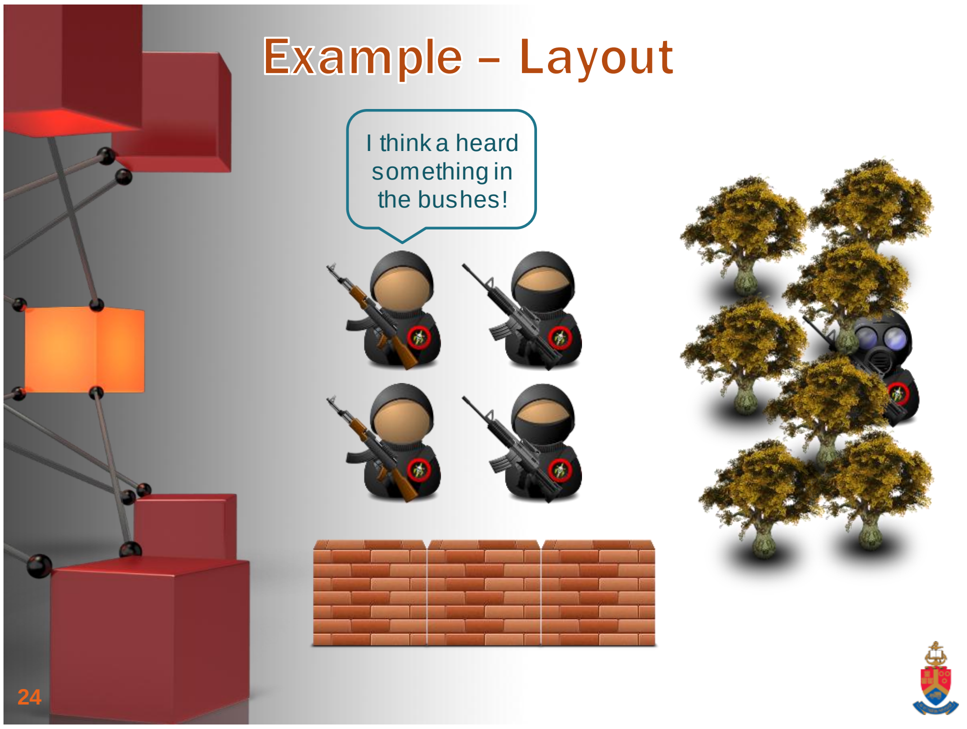


Example – Layout

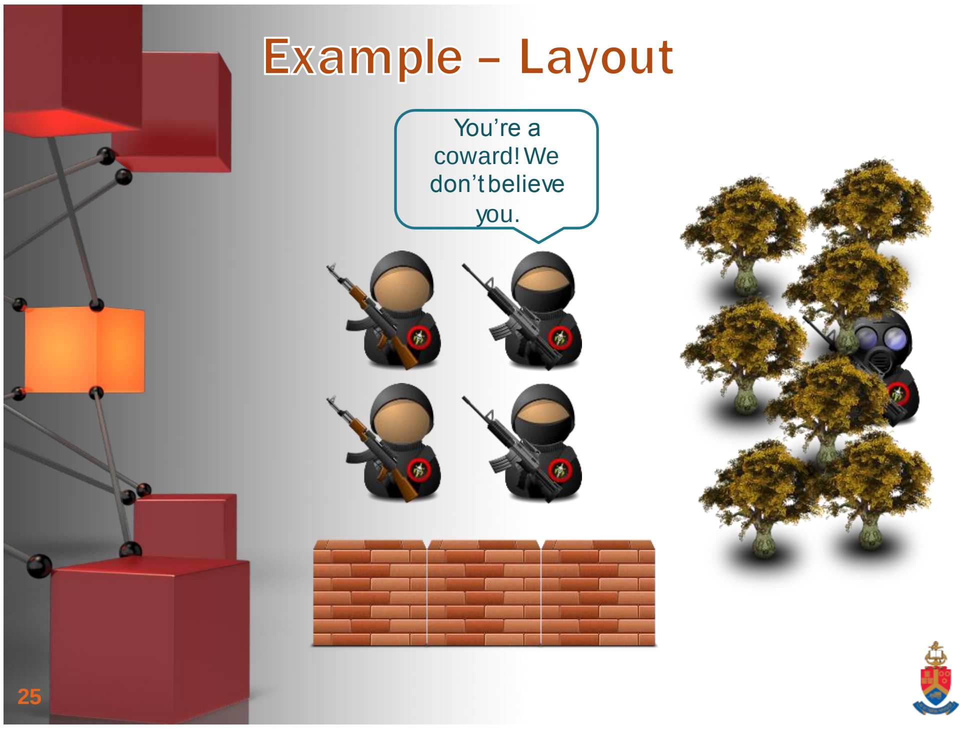
22



Example – Layout

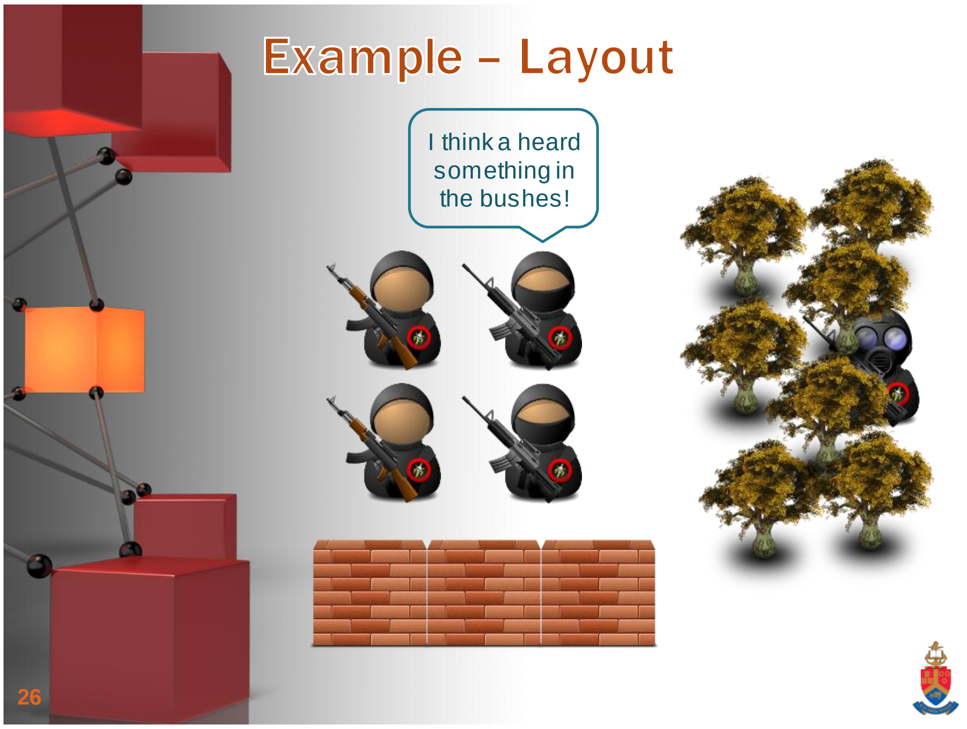


Example – Layout



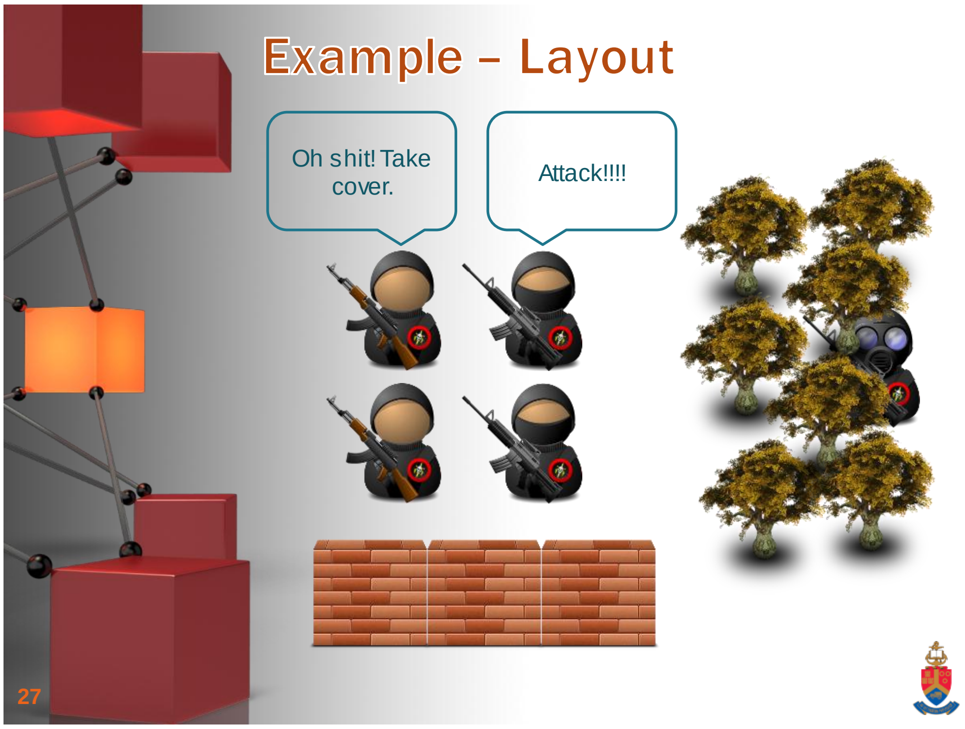
Example – Layout

25

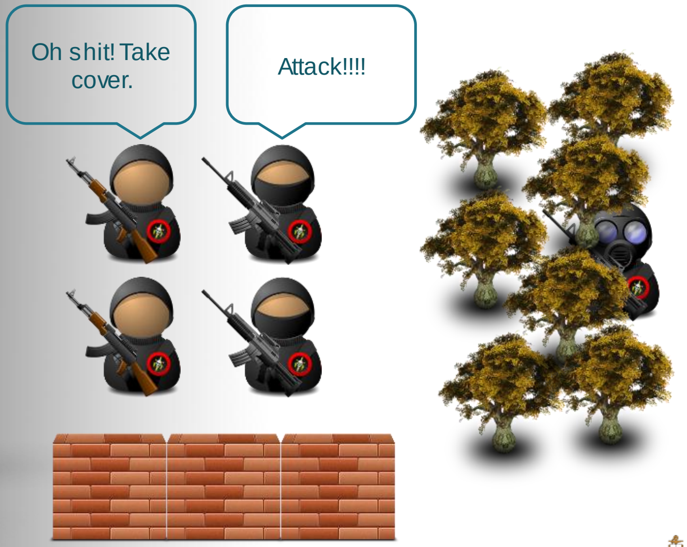


Example – Layout

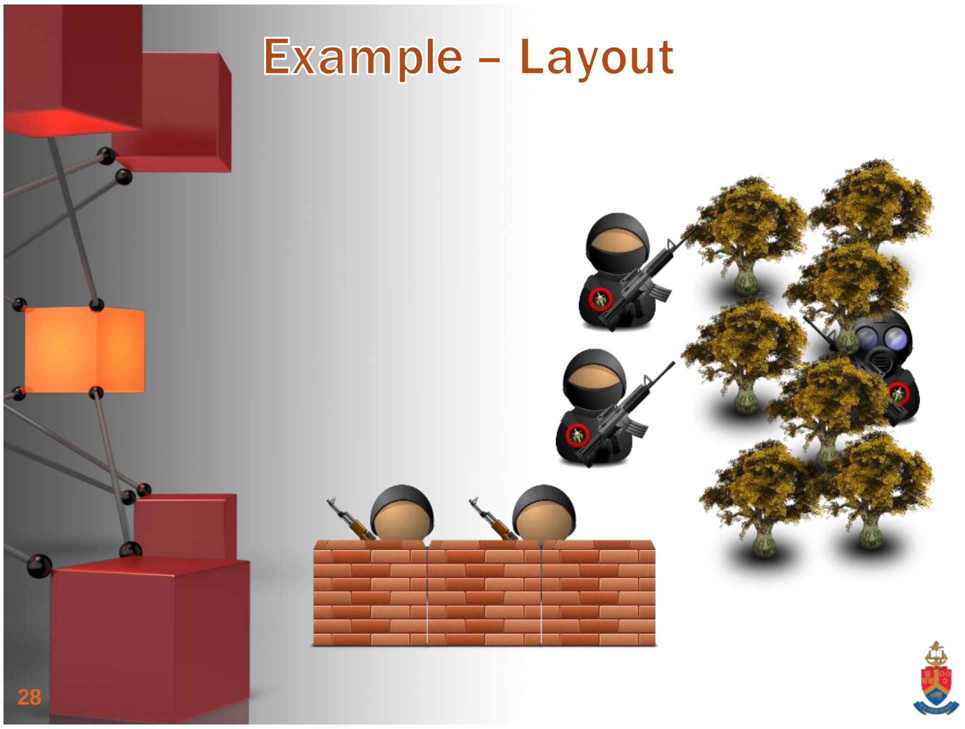
26



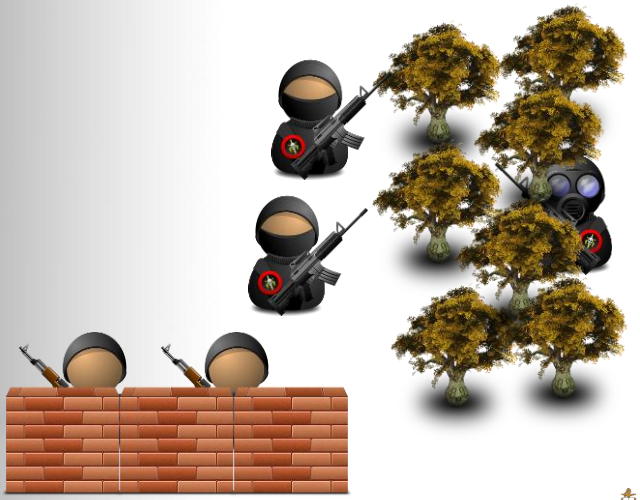
Example – Layout



27

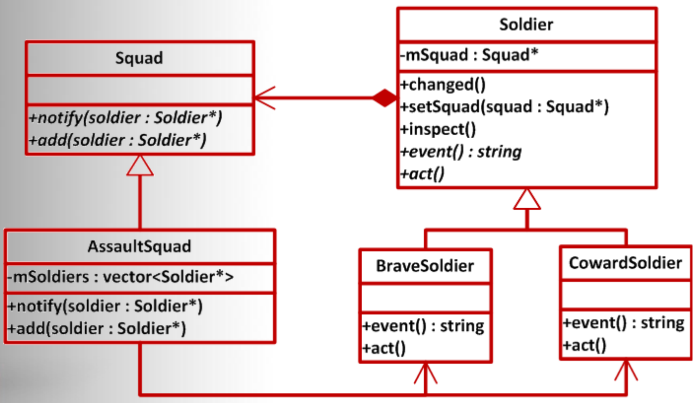


Example – Layout



28

Example – UML

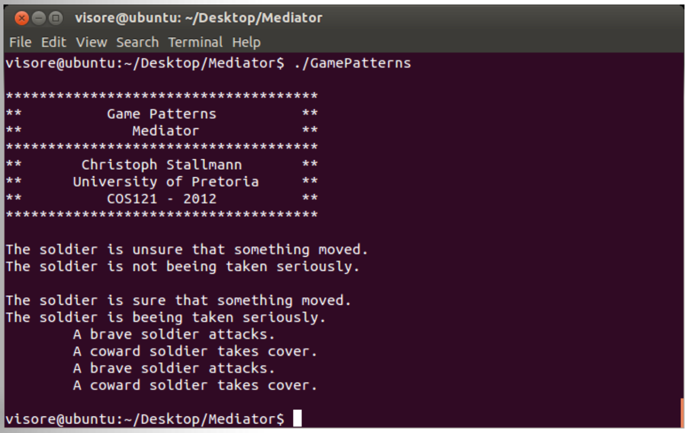


29

Example – Code

30

Example – Output


visore@ubuntu: ~/Desktop/Mediator

File Edit View Search Terminal Help

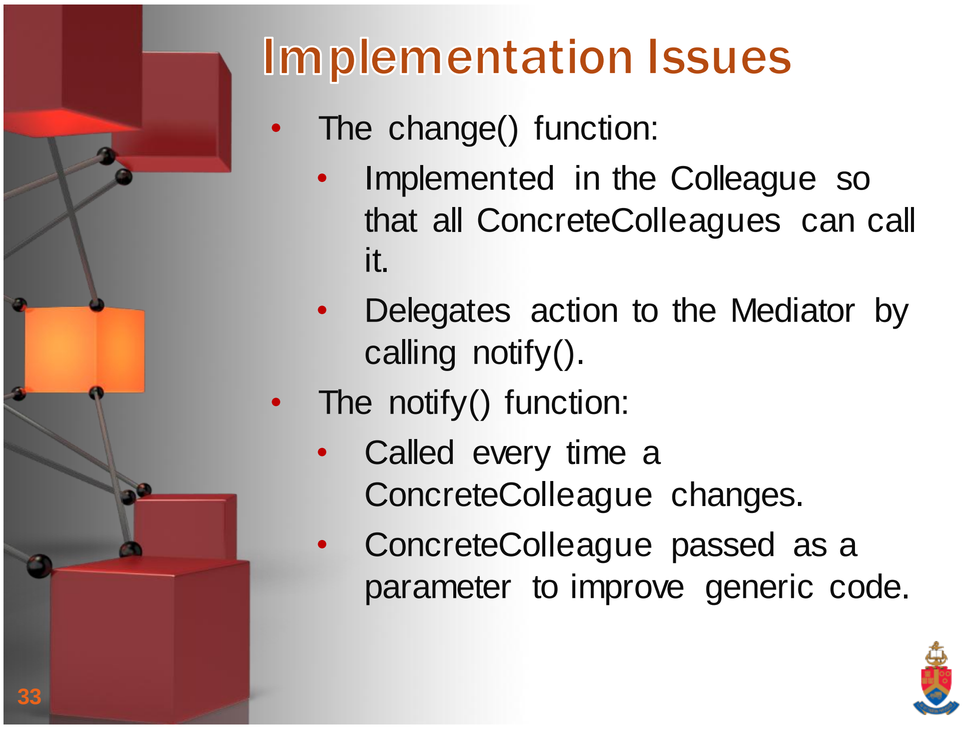
```
visore@ubuntu:~/Desktop/Mediator$ ./GamePatterns

*************************************
**          Game Patterns          **
**             Mediator            **
*************************************
**       Christoph Stallmann       **
**      University of Pretoria     **
**          COS121 - 2012          **
*************************************

The soldier is unsure that something moved.
The soldier is not beeing taken seriously.

The soldier is sure that something moved.
The soldier is beeing taken seriously.
        A brave soldier attacks.
        A coward soldier takes cover.
        A brave soldier attacks.
        A coward soldier takes cover.

visore@ubuntu:~/Desktop/Mediator$ 
```

31

Improvements Achieved

- Simplification of code updates:
  - Without the Mediator, all classes have to be updated to accommodate a code change.
- Increased reusability of code:
  - Decoupling of Colleagues increases the individual cohesiveness and improves reusability.
- Simplification of object protocol:
  - One-to-many replaces a many-to-many relationship which is easier to understand and maintain.

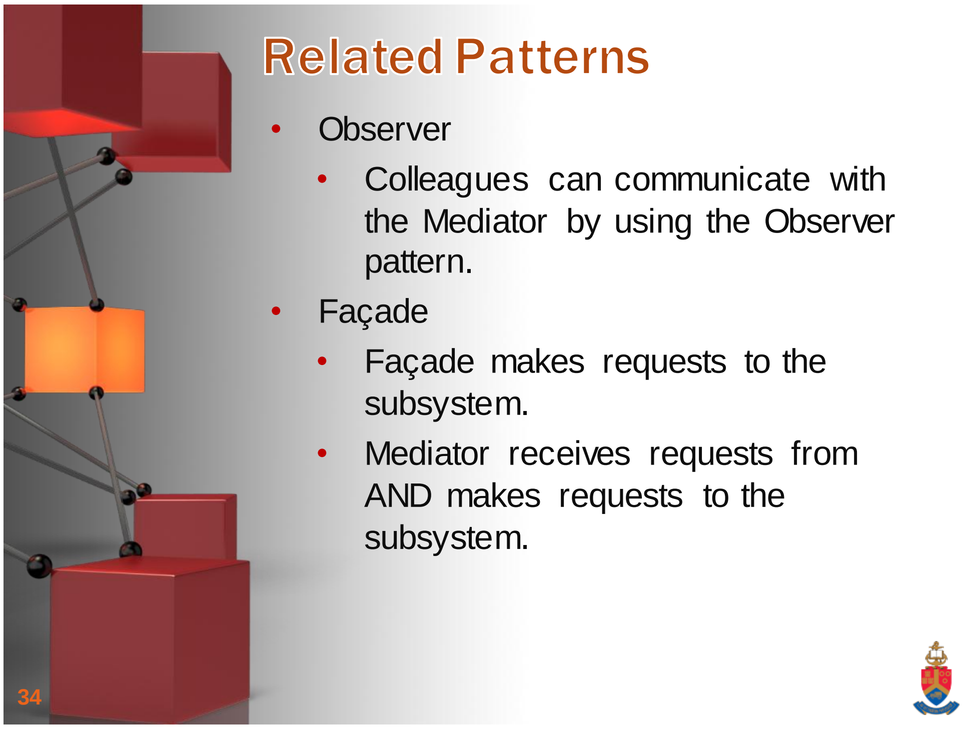
32



Implementation Issues

- The change() function:
  - Implemented in the Colleague so that all ConcreteColleagues can call it.
  - Delegates action to the Mediator by calling notify().
- The notify() function:
  - Called every time a ConcreteColleague changes.
  - ConcreteColleague passed as a parameter to improve generic code.

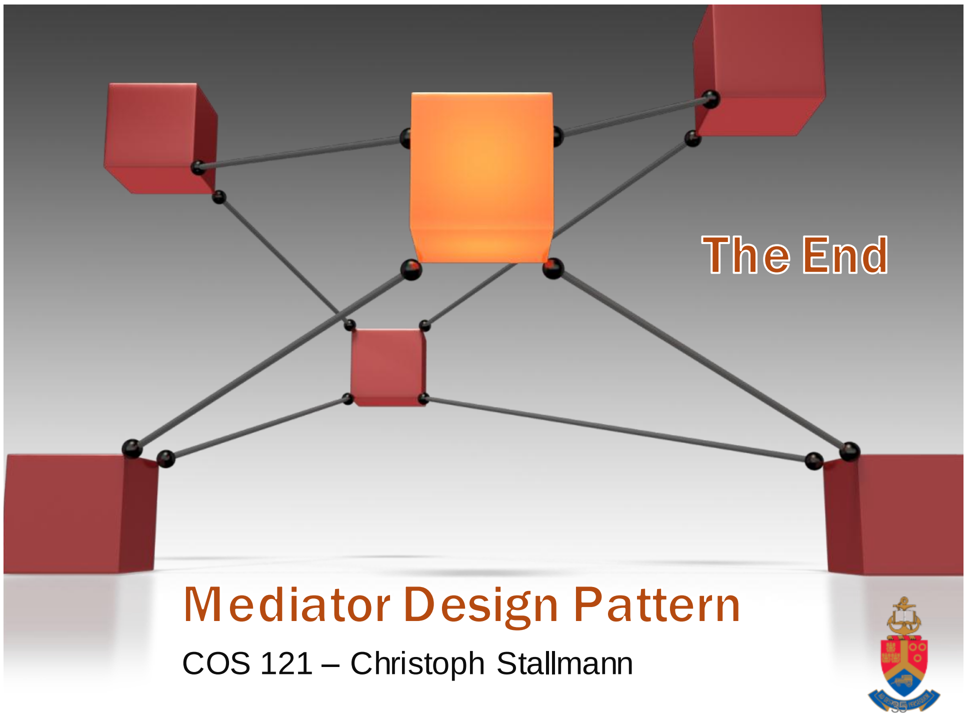
33



Related Patterns

- Observer
  - Colleagues can communicate with the Mediator by using the Observer pattern.
- Façade
  - Façade makes requests to the subsystem.
  - Mediator receives requests from AND makes requests to the subsystem.

34



The End

Mediator Design Pattern

COS 121 – Christoph Stallmann