Observer

Sandro Santos Andrade

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas

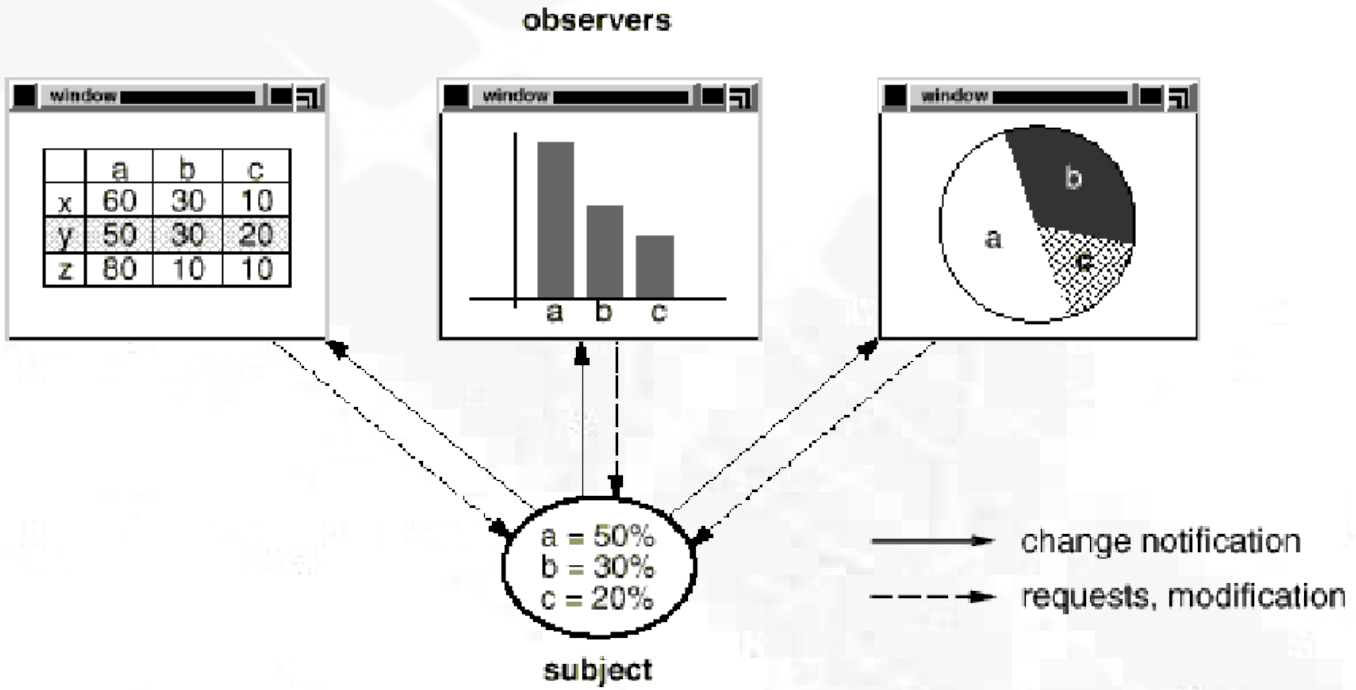
Observer

-
  -
-
-
  -

Dependents Publish-Subscribe

Observer

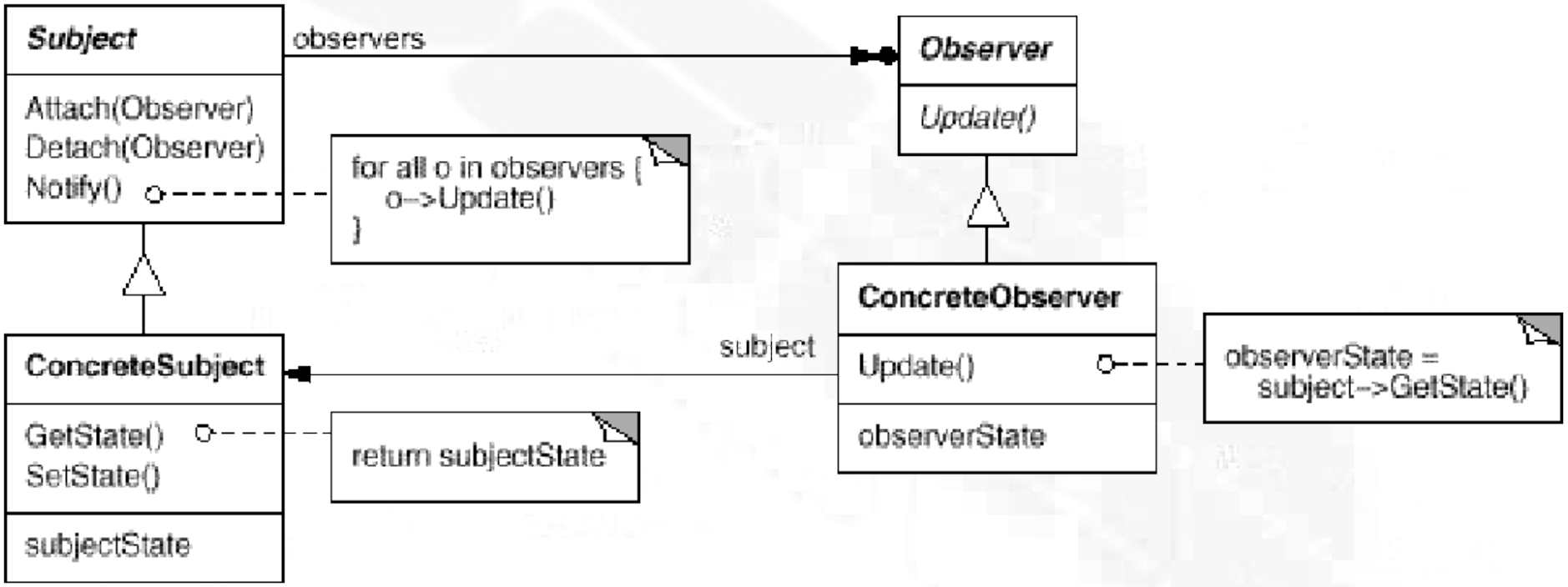
Observer



- *subject observers*
- *Publish-Subscribe: observers subject(s)*

Observer

Observer


Subject
observers
Attach(Observer)
Detach(Observer)
Notify()
for all o in observers {
o->Update()
}
Observer
Update()
ConcreteSubject
GetState()
SetState()
subjectState
return subjectState
subject
ConcreteObserver
Update()
observerState
observerState =
subject->GetState()

Observer

-
  - *Subject*
    - *observers* *observers*
      subject
    - *observers*
  - *Observer*
    -
      subject

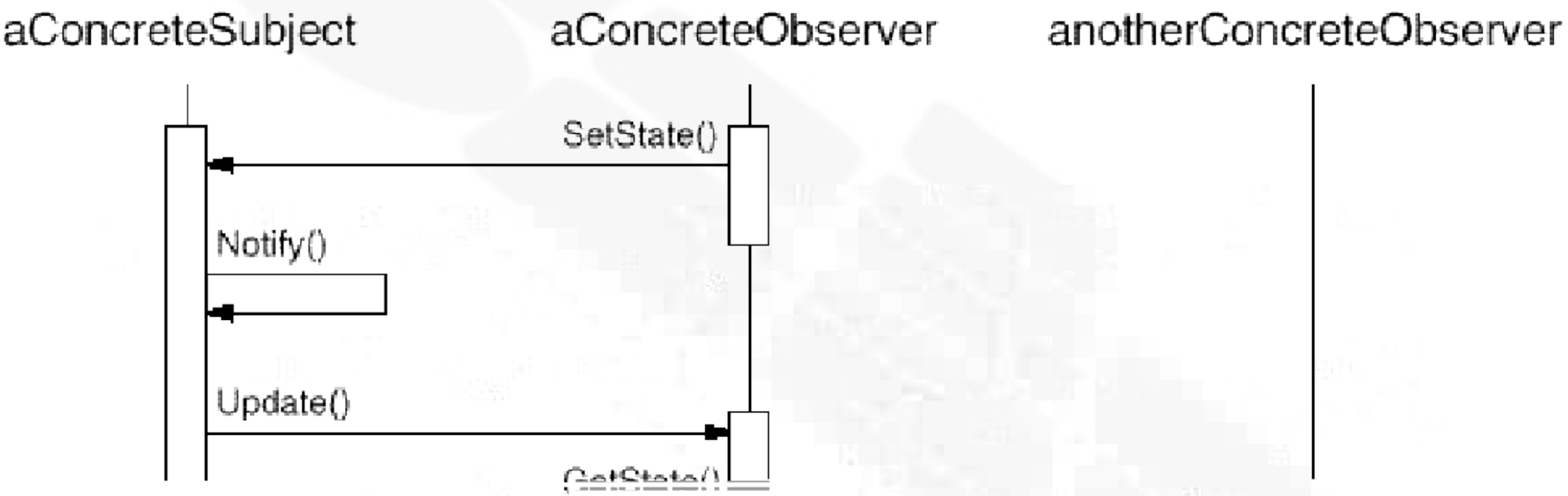
Observer

-
  - *ConcreteSubject*
    - *ConcreteObservers*
    - *observers*
  - *ConcreteObserver*
    - *ConcreteSubject(s)*
    - *subject*
    - *Observer*

Observer

-
  - *ConcreteSubject* *observers*
    observer
  - *ConcreteSubject* *ConcreteObserver*
    ConcreteSubject
    ConcreteSubject

Observer


aConcreteSubject
aConcreteObserver
anotherConcreteObserver
SetState()
Notify()
Update()

Observer

-
  - *Subject* *Observer*
    - *subject* *observers*
    - *subject* *observer*
  - *broadcast*
    - *run-time*

Observer

-
  -
    - *observers*
    - *observers*
    - *subject*
    - *subject*
    -
    -
    - *observers*

Observer

-
  - *subjects* *observers*
    - *subject* *observers*
    - *subjects* *observers*
    - *look-up* *Subjects* *observers* *overhead* *observers*

Observer

-
  - *subject*
    - *observer* *subject*
    - *subject*

Observer

-
  -
    - *notify()*
      -
      -
    -
      - *subject*
      -

subject

notify()

Observer

-
  -
    - *subject*
      observers
    - *subjects*
      observers
    - *observers*
      subjects

Observer

-
  - *subject*
    - *observers* *subject(s)*
    -

```
void MySubject::Operation (int newValue) {
BaseClassSubject::Operation(newValue);
// trigger notification
_myInstVar += newValue;
// update subclass state (too late!)
}
```

Observer

-
  - *subject*
    - *template methods*

```
void Text::Cut (TextRange r) {
ReplaceRange(r);        // redefined in subclasses
Notify();
}
```

Observer

-
  - *observer*
    - *subject*

      update()
      - **Modelo Push** *subject* *observer*
        - *subjects*
      - **Model Pull** *subject* *update()* *observers* *subject*
        -

Observer

-
  -
    -

```
void Subject::Attach(Observer*, Aspect& interest);




void Observer::Update(Subject*, Aspect& interest);
```

Observer

-
  -
    -
    -
      -
      -
      -
      -
      -

ChangeManager

ChangeManager

subject *observers*

Subjects

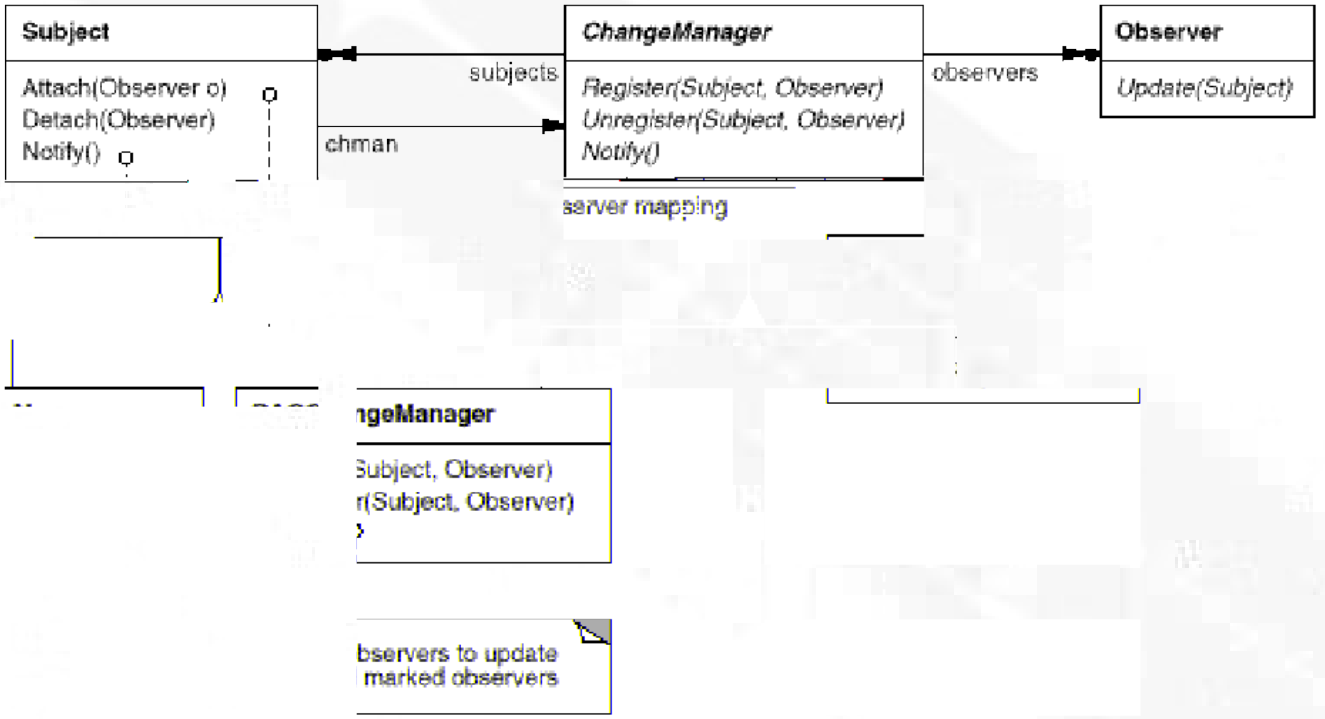
observers

subject,

observers

Observer

-
  - *ChangeManager*



  - *mediator* *singleton*

Observer

-
  - *Subject* *Observer*
    -

Observer

```
class Subject;


class Observer {
public:
        virtual ~ Observer();
        virtual void Update(Subject* theChangedSubject) = 0;
protected:
        Observer();
};
```

Observer

```
class Subject {
public:
        virtual ~Subject();
        virtual void Attach(Observer*);
        virtual void Detach(Observer*);
        virtual void Notify();
protected:
        Subject();
private:
List<Observer*> *_observers;
```

```
void Subject::Attach (Observer* o) {      _observers->Append(o);    }

void Subject::Detach (Observer* o) {      _observers->Remove(o);    }

void Subject::Notify () {
ListIterator<Observer*> i(_observers);
for (i.First(); !i.IsDone(); i.Next()) {
        i.CurrentItem()->Update(this);
}
}
```

Observer

```
class ClockTimer : public Subject {
public:
        ClockTimer();
        virtual int GetHour();
        virtual int GetMinute();
        virtual int GetSecond();
        void Tick();
};

void ClockTimer::Tick () {
        // update internal time-keeping state
        // ...
        Notify();
}
```

Observer

```
class DigitalClock: public Widget, public Observer {
public:
        DigitalClock(ClockTimer*);
        virtual ~DigitalClock();
        virtual void Update(Subject*);
                // overrides Observer operation
        virtual void Draw();
                // overrides Widget operation;
                // defines how to draw the digital clock
private:
        ClockTimer* _subject;
};
```

Observer

```
DigitalClock::DigitalClock (ClockTimer* s) {
        _subject = s;
        _subject->Attach(this);
}

DigitalClock:: DigitalClock () {
        _subject->Detach(this);
}

void DigitalClock::Update (Subject* theChangedSubject) {
        if (theChangedSubject == _subject) {
                Draw();
        }
}

void DigitalClock::Draw () {
        // get the new values from the subject

        int hour = _subject->GetHour();
        int minute = _subject->GetMinute();
        // etc.

        // draw the digital clock
}
```

Observer

```
class AnalogClock : public Widget, public Observer {
public:
        AnalogClock(ClockTimer*);
        virtual void Update(Subject*);
        virtual void Draw();
        // ...
};
```

```
ClockTimer* timer = new ClockTimer;
AnalogClock* analogClock = new AnalogClock(timer);
DigitalClock* digitalClock = new DigitalClock(timer);
```

Observer

-
  - *ChangeManager Mediator subjects observers*
  - *ChangeManager Singleton*

Observer

Sandro Santos Andrade

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas