Design Patterns

MSc in Communications Software

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Further GUI Patterns

Sources & Context

Separated Presentation

- A form of layering where presentation code and domain code in separate layers with the domain code unaware of presentation code.
    - Review all the data and behavior in a system identifying code involved in the presentation. Presentation code would manipulate GUI widgets and structures only.
    - We then divide the application into two logical modules with all the presentation code in one module and the rest in another module.
- The layers are a logical and not a physical construct. Java packages are a useful separation mechanism

Separated Presentation - Synchronization

- It will be necessary for the domain to notify the presentation if any changes occur.

- 2 common solutions:

  - Flow Synchronization

  - Observer Synchronization

- Both of these solutions attempt will ensure that the model and the views are rendering the same information.

- However, both must ensure that Separated Presentation is preserved.

Model View Controller

- The Model/View/Controller (MVC) triad of classes is used to build user interfaces in Smalltalk-80.

- MVC consists of three kinds of objects:

  - Model is the application object

  - View is its screen presentation

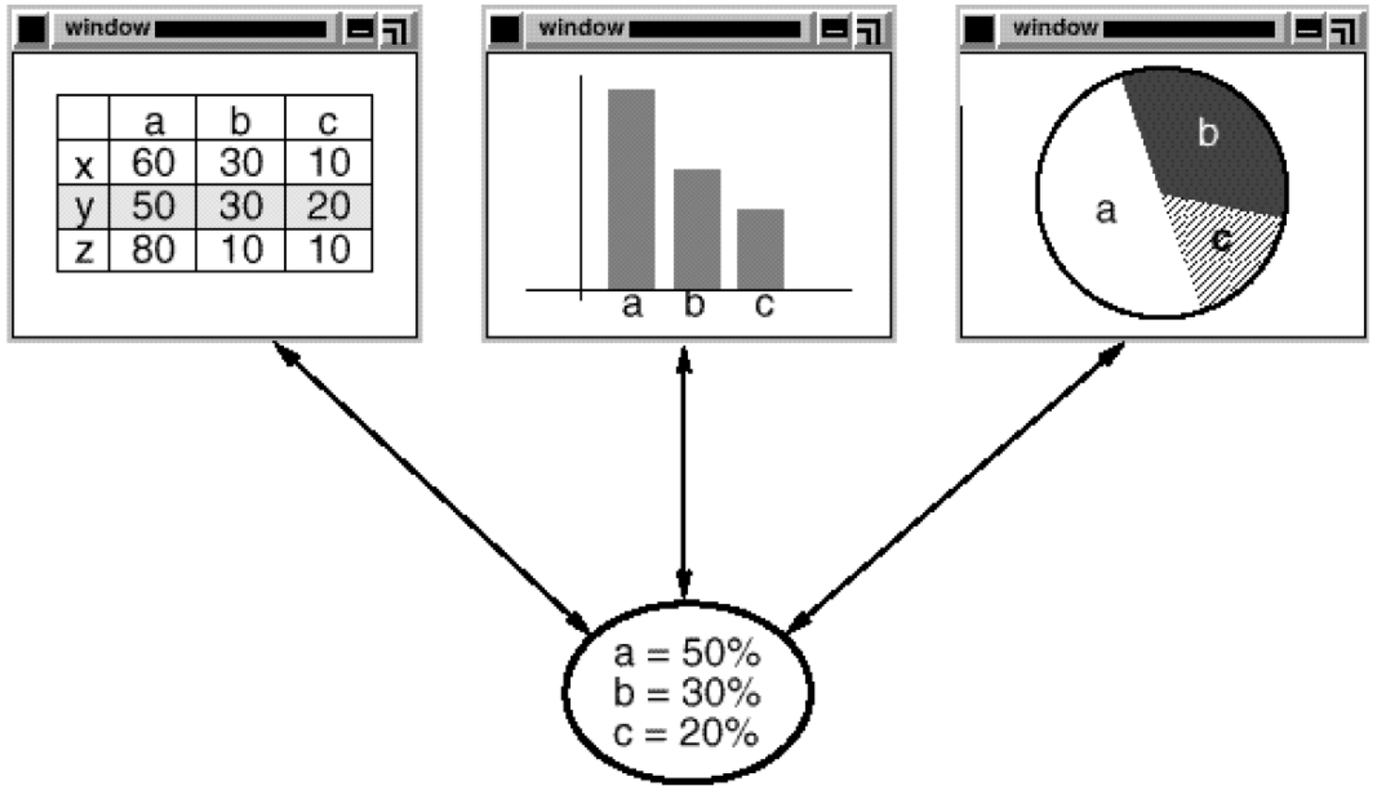
  - Controller defines the way the user interface reacts to user input

- Before MVC, user interface designs tended to lump these objects together. MVC decouples them to increase flexibility and reuse

Synchronization

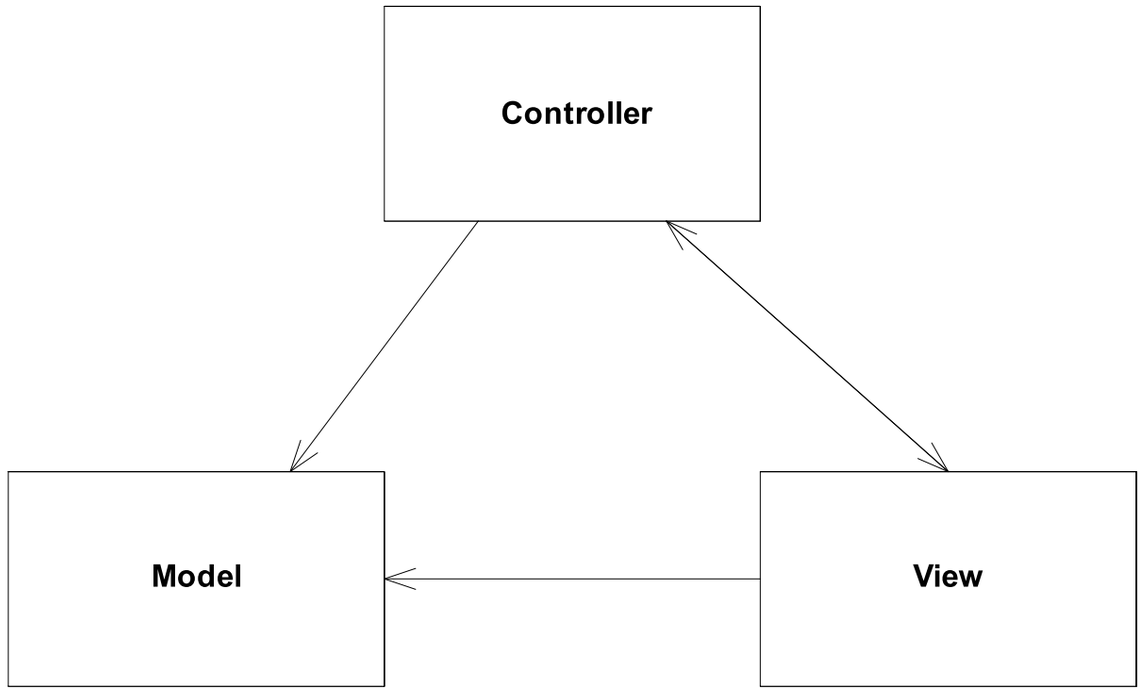
- MVC synchronizes views and models via ObserverSynchronization.
    - A view must ensure that its appearance reflects the state of the model.
    - Whenever the model's data changes, the model notifies views that depend on it.
    - In response, each view gets an opportunity to update itself.
- This approach allows multiple views to be attached views to a model to provide different presentations.

View / Model


window
a b c
x 60 30 10
y 50 30 20
z 80 10 10
window
a b c
window
b
a
c
a = 50%
b = 30%
c = 20%

Controller

- MVC encapsulates response and model update mechanisms in a Controller object.
- The Controller is the “glue” between the Model and the View.
- The View renders model updates on the screen, but is not permitted to modify the model.
- The View forwards events to the controller
- The Controller does not have access to the screen but can modify the model.



Patterns in MVC: A “Composed” Pattern

- Observer:
  - Decouple objects so that changes to one can affect any number of others without requiring the changed object to know details of the others
  - The pattern that decouples views from models.
- Strategy:
  - An object that represents an algorithm. Used when an algorithm is to be replaced either statically or dynamically, or when the algorithm has complex data structures that you want to encapsulate.
  - The pattern encapsulates the View-Controller relationship- the Controller is a strategy used by the View.
- Other Patterns:
  - Composite: Views can be nested. E.g. a panel of buttons and tables can be implemented as a view containing “composite” button views and table views.
  - Façade: the Model may in fact be a Façade to a more complex model.

MVC & Web Frameworks

- MVC has been successfully adapted into web application domain.
- MVC implementations are central to Spring, Wicket, Java Server Faces and a large number of similar frameworks...
- In Rich Clients (GUI) applications it has been less successful...
- see [http://martinfowler.com/eaaDev/uiArchs.html] for a more detailed history of MVC.

Model View Presenter (MVP)

- Can be considered to be a updating of MVC for modern GUI development.
- Also a “Composed” pattern:
    - Separated Presentation
    - Observer Synchronization
    - Passive View
    - Supervising Controller

Passive View

- View is made completely passive and is no longer responsible for updating itself from the model.

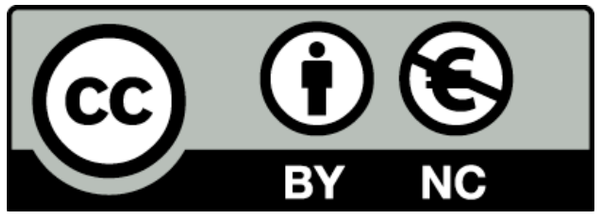
- Views are simply painted by a GUI Designer, and accessors are provided to retrieve the controls for initialization elsewhere.

- This leaves the view completely “passive” - no initialization, no event handlers no interesting behaviour of any kind.

- All of this behaviour is moved to an associated Supervising Controller/ Presenter.

Supervising Controller/Presenter

- A Supervising Controller has two primary responsibilities: input response and partial view/model synchronization.
    - For input response the user gestures are handled initially by the screen widgets, however all they do in response is to hand these events off to the controller/presenter, which handles all further logic.
    - For model/view synchronization, the controller/presenter can employ ObserverSynchronization.



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit