MVC and Friends

Joshua Charles Campbell

Department of Computing Science
University of Alberta

Febuary 2014

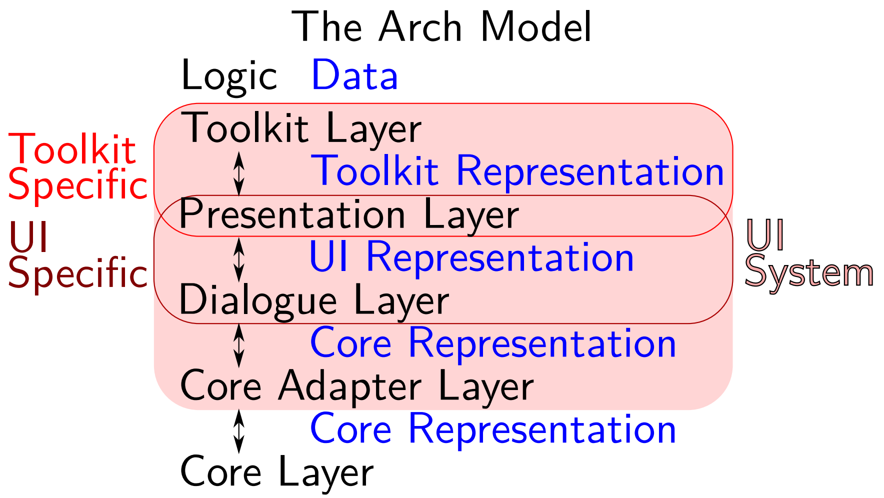
Outline

1 Introduction

2 Modern MVC

3 Related Patterns

Arch Model


The Arch Model
Logic
Data
Toolkit Specific
UI Specific
Toolkit Layer
Toolkit Representation
Presentation Layer
UI Representation
Dialogue Layer
Core Representation
Core Adapter Layer
Core Representation
Core Layer
UI System

Core

- Formal Domain **Data Represenations**
  - Economical and Unambiguous
  - What you would serialize
- Informal Domain Data Representations
  - May contain redudant data, non-canonical forms, etc.
  - Think "non-normalized"

Core cont.

- **Data representations** about the fundamental "things" the application is working with
- Logic enforcing **data constraints**
    - Prevent **data representations** which are invalid in the domain

Core cont.

- Logic relating multiple **core data representations**
  - Model Evolution
  - Converting to other fundamental representations
- This forms the *Functional Core* in the Arch Model

Core Adaptor

- Logic and data provided for the use of any/multiple user interfaces
- Connects **user interfaces** to the **core** using the **core data representations**

Dialogue Component

- All of the *UI-specific* but *toolkit-independent* **data represenations** and **logic**
- May contain all kinds of stuff that the core wouldn't
  - Application States, feedback for the user, redundant data forms
  - Sequencing and consistency logic

Dialogue Component

- Gets **core data representations** from the **core / core adapter**
- Sends **toolkit-independent data representations** to the **presenter**
- Fowler calls this the *presentation model*

Presenter

- Format data to be passed to the toolkit library
- Format data in a **toolkit-independent** form for the dialogue
- Interfaces the **toolkit** to **dialogue**
- *Presentation Component* of the Arch Model

The Interaction Toolkit

- Knows nothing about the model or the user interface
- Checkboxes, scrollbars, windows, layouts, picture boxes, etc.

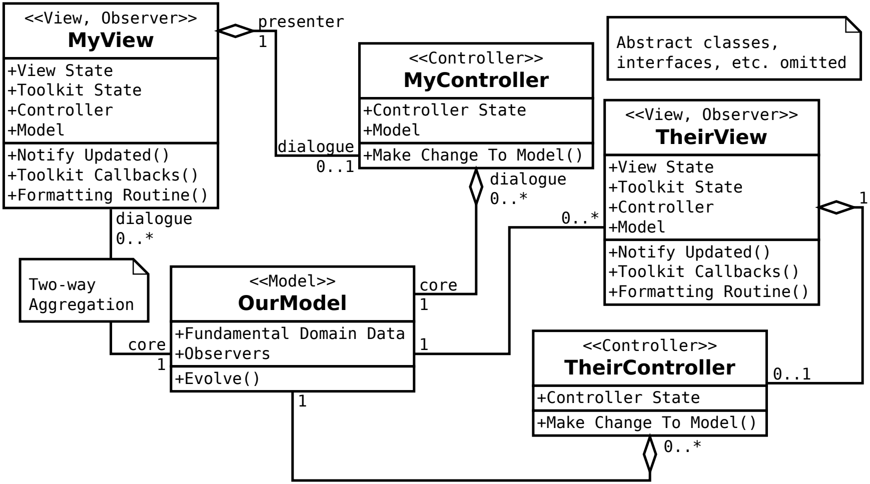
Model-View-Controller

- Every View must have a reference to a controller and a model
- Every Controller must have a reference to a model
- Multiple View-Controller pairs may share a single model simultaneously

Active Model MVC

- Recommended
- The model has a reference to views needing update
- aka Observer Synchronization

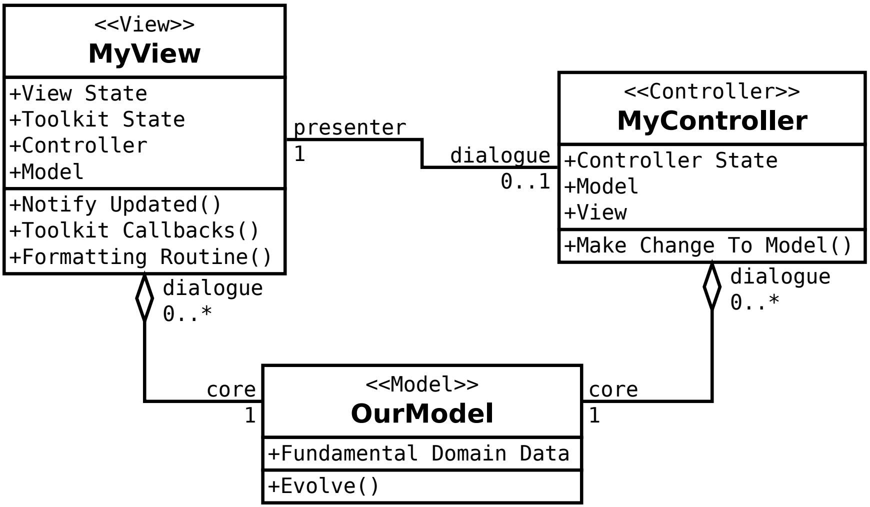
Active Model MVC



Passive Model MVC

- Not recommended
- The controller has a reference to views needing update
- aka Flow Synchronization

Passive Model MVC



Model-View-Controller

According to most modern sources:

- Model contains the **core**, **core adapter**
  - logic to enforce consistency
  - logic to enforce sequencing
- Controller contains **presenter** and **dialogue** input
  - All logic that interprets user actions as modifications for the model
- View has **presenter** and **dialogue** output
  - All logic that makes the model ready for the toolkit
- We get the actual toolkit from someone else

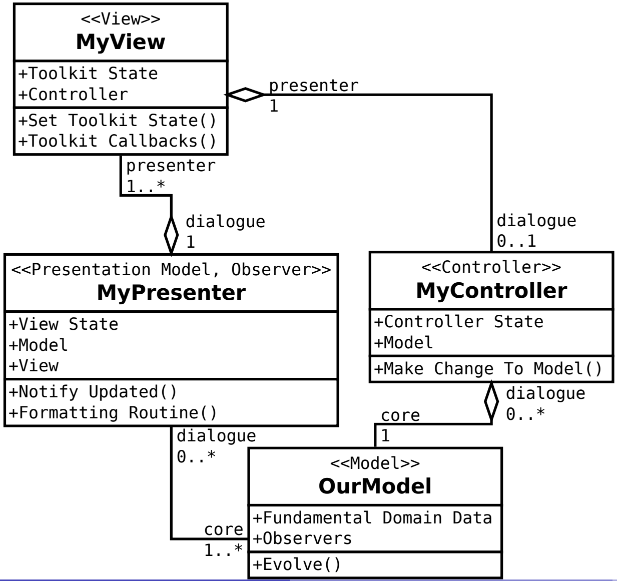
Model-View-Controller cont.

- Model must support
  - multiple, independent view-controller pairs
  - which are completely ignorant of each-other
  - possibly at the same time
- Problem: leads to duplicated **dialogue** code
  - Ex: view state

Presentation-Model MVC

- MVC except:
    - Split the view and controller into toolkit-specific and toolkit-independent parts
    - Fixes duplication problem

Presentation-Model MVC



MVC "Classic"

- MVC ala Smalltalk 80
- Like MVC but V-C pairs implement the toolkit also
- V-C pairs contain
    - The **toolkit** code & data
    - **Toolkit-specific** code & data

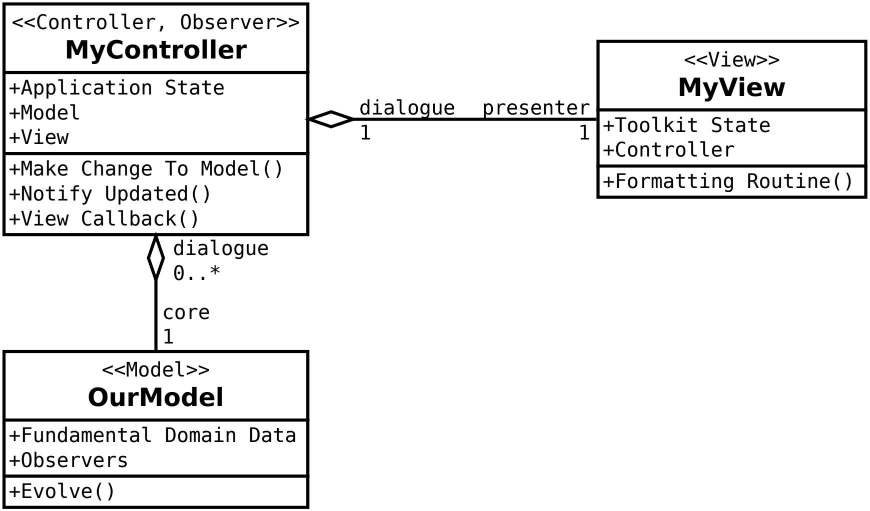
MVC "Classic" cont.

- Model contains
    - The **Dialogue**
    - All of the **core** and **core adapter**
    - Application state, sequencing, consistency, feedback for the user, etc.

Passive-View MVC

- Model and View completely disconnected
- View is as light and generic as possible
- Controller connects the Model to the View
- all **dialogue** is in the controller
  - All application-specific logic
- what *Ruby on Rails* thinks of as MVC

Passive-View MVC



Interface-Control-Model

- The *model* is the **core**
  - Plus a list of things to notify on change
- The *control layer* consists of the **dialogue**
  - Much app code goes here
- The *interface layer* is **toolkit-specific**
  - Unlike View, can recieve commands in order to pass them to the control layer in a toolkit-independent way

Model-View-Presenter

- Much like Passive View
- Adds: Commands, selections and interactors

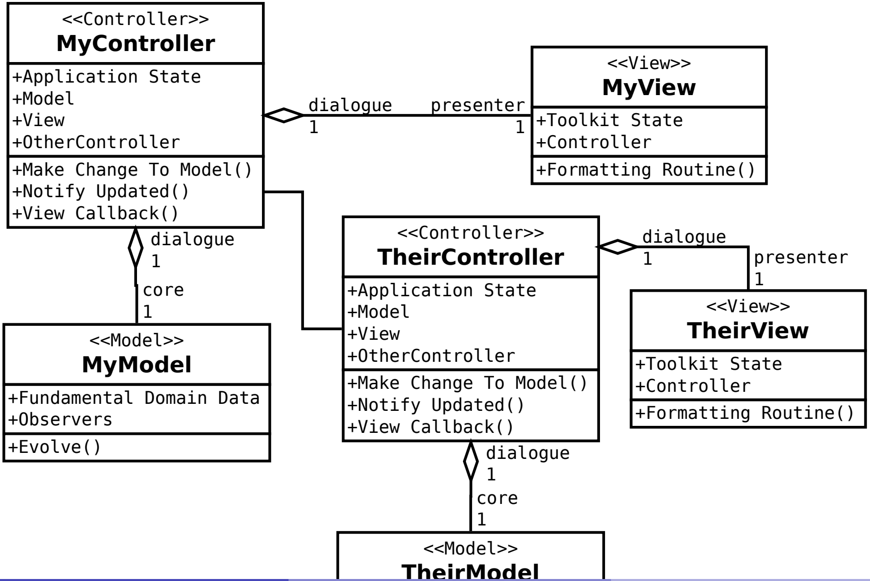
Presentation-Abstraction-Control

- Everything is connected via a heirarchy of controllers
- Models are connected to other models via the controller
- Views are connected to other views via the controller
- Two views or controllers using the same model at the same time is disallowed

Presentation-Abstraction-Control

- Views are toolkit-specific **presenters** and **toolkits** only
- Models are **core** only
- All **dialogue** is in the controller
  - Including output, display, formatting, etc. logic
- Like a bunch of passive-view MVCs connected by their controllers
  - instead of their models
- what *Stanford University* of as MVC

Presentation-Abstraction-Control



Conclusion

- MVC often refers not to MVC but to
  - Passive-view, PAC, ICM, or other systems where a lot of logic that would be in the model or view in MVC is completely inside the "controller"
- MVC almost never refers to the original, pixels-and-cursors MVC

Bibliography I

Steve Burbeck.
Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC).
Smalltalk-80 v2, 5, 1992.

Frank Bushmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal.
Pattern-oriented software architecture: A system of patterns.
John Wiley&Sons, 1996.

Kyle Brown C2 Wiki.
What's a Controller, Anyway?, 2013.

Martin Fowler.
GUI Architectures, 2006.

Martin Fowler.
Passive View, 2006.

Martin Fowler.
Presentation Model, 2006.

Paul Hegarty.
MVC and Introduction to Objective-C, 2011.

Bibliography II

Greg Hendly and Eric Smith.
Seperating the gui from the application.
The Smalltalk Report, 1(7):19–22, 1992.

Panagiotis Markopoulos.
A compositional model for the formal specification of user interface software.
PhD thesis, University of London, 1997.

Mike Potel.
Mvp: Model-viewer-presenter, 2000.

C2 Wiki.
Model View Controller, 2013.

Wikipedia.
Presentation-abstraction-control — wikipedia, the free encyclopedia, 2014.
[Online; accessed 21-February-2014].