1.3 The MVC and MVP Patterns

The ancestors to the MVVM pattern are the MVC and MVP patterns, so I will present them before I dig into the MVVM pattern in the next chapter. If you are already familiar with these patterns you may skip this section.

1.3.1 Model View Controller (MVC)

Origin

The Model View Controller design pattern (MVC) was developed by the Norwegian professor Trygve Reenskaug during his stay at Xerox Parc in 1978 – 1979. The group Reenskaug worked with experimented with development of graphical user interfaces and MVC pattern was conceived as a general solution to the problem of users controlling a large and complex data set (Reenskaug 2003) and the capability to have several views visualizing the same data set.

Jim Althoff and others implemented a version of MVC for the Smalltalk-80 class library after Reenskaug had left Xerox Parc, and they used the term Controller somewhat differently from Reenskaug's original idea (Reenskaug 2003). A description of this implementation can be found in the work of S. Burbeck (Burbeck 1992) and this is the version I will present here.

A more thorough description of the MVC pattern can be found in book Pattern-Oriented Software Architecture 1 page 125 (Buschmann, Meunier et al. 1996)

Structure

Figure 1 shows the structure of Model-View-Controller pattern as implemented in the Smalltalk-80 class library. This figure is a simplistic visualization of the principal relationships in the MVC pattern. In a real application there will typical be many different View classes, Controller classes and Model classes.

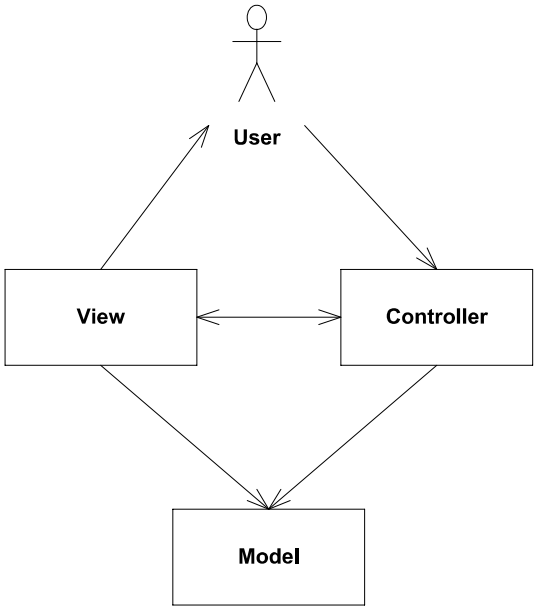

Figure 1: The structure of the MVC pattern

Model

The model holds the data when the application is running, and also contains the business logic. How the model is organized internally is free for the developer to decide. But it is mandatory that the model does not depend on the View or Controller. If the Model has to update the View or Controller this must be done by use of the Observer Design Pattern (or a similar notification mechanism).

View

The view is responsible for showing the model to the user – or more often: it will show certain parts or properties of the model on the screen (GUI). A View typical only has a minor area of the screen real estate as its output area, but it may own the whole screen. Several Views can be connected to the same model – each View will visualize different properties of the model, or visualize the data in different ways (e.g. as numbers or as a bar chart). The View is dependent on both the model and its associated controller. The View does not send a lot of messages to its Controller, but the View creates and releases its Controller.

Controller

The Controller gets all input from the user such as keyboard stokes and mouse. Depending on the kind of input the Controller gets it will update the model or send commands to its associated View. When the controller updates the model, the View will be notified from the Model through the observer design pattern. In an application there will be one Controller for each View, but only one Controller will receive input at any one time.

Key concepts

A strong separation between the presentation layer (View and Controller) and the business logic layer (Model) which facilitates reuse of the model in new presentation frameworks. This is very important as time has shown that presentation frameworks changes more rapidly than business logic.

Use of the Observer design pattern makes it possible to have multiple Views connected to the same Model and still maintain a loose coupling in the application.

Variations

The MVC pattern in its original form as presented above is seldom used today, but there are many variants of sthe original pattern that people still refer to as the MVC pattern – sometimes quite misleading.

Document-View:

In this variant the View and Controller is collapsed to one class. Most modern GUI frameworks include a lot of generic widgets (called controls on the Windows platform) that both handle output and input and thereby make the original Controller in MVC obsolete.

The Web-based MVC pattern:

Many Web server frameworks use the MVC-pattern, but they typical have the Controller as the main component and responsible for creation of the appropriate View, and they may have a front-controller responsible for handling requests over to the relevant controller (Greer 2007). Sun's Model 2 is the first well known Web server framework that follows the MVC-pattern (Seshadri 1999), but there are several others e.g. ASP.Net MVC.