MVC

It allows you to make your program more modular. Programmers might use the term loosely coupled. Essentially, you are breaking your program into separate pieces that don't rely upon each other to function. The idea is that you can switch out each piece without affecting the others. The pieces are "M" for Model layer which contains classes used to create objects (i.e. a student, a car, etc.). The "V" is for View layer, which is the interface to the program. This is how you program will receive input and send output. Finally, there is "C" for Controller layer, which controls the flow of data from the view layer to the model layers.

A good example is if you had an application that was built using forms for the UI, and the MVC design pattern was used, then you could easily change the UI to a web based UI.

[https://github.com/hord-brayden/CIT-360/tree/master/14 Topics/Afghan-Trail-master/Afghan\_Trail/src/byui/cit260/afghan\_trail](https://github.com/hord-brayden/CIT-360/tree/master/14%20Topics/Afghan-Trail-master/Afghan_Trail/src/byui/cit260/afghan_trail)

MVC is a good process to adopt. There are a couple of other design ideas, but as I continue to implement it, I find that it works pretty well.

One of the principles I like, is decoupling of the View. This is because the View (what the user sees and interacts with) can change so frequently. The view could be a web page, an android or iphone interface, or a windows universal app. All of them simply need to communicate with the controller.

The controller does all the heavy middle man stuff. Long long time ago, I used to do all the work on the same page or form that the user interfaced with. When that format changed, it was a huge problem.

Now I can simply ask for an object, the controller contacts the model, the model might retrieve it from the database (or wherever) then passes it back to the controller and the controller serves it up to the view.

Rick

MVC is a schema of organization. It's just separating different features and functions into the model, view, and controller folders respectively.

Here's my link to my MVC, [https://github.com/hord-brayden/CIT-360/tree/master/14%20Topics/Afghan-Trail-master/Afghan\_Trail/src/byui/cit260/afghan\_trail (Links to an external site.)Links to an external site.](https://github.com/hord-brayden/CIT-360/tree/master/14%20Topics/Afghan-Trail-master/Afghan_Trail/src/byui/cit260/afghan_trail)

Each one of the three serves a functional purpose. Here's a VERY SIMPLIFIED version of what they do!

**The View** is what the user sees, **the Controller** handles input, and **the Model** contains the background data being used and calculations.

I personally only started using it when working with a team, because it keeps everyone on the same page with organization and contributions.

<https://alvinalexander.com/uml/uml-model-view-controller-mvc-diagram>