I picked the MOVE (Models, Operations, Views, Events) pattern to discuss because I had not heard of it before. My research results seem to indicate that the MOVE pattern never gained the popularity of the MVC or MVVM patterns. Very few mentions are found on the web between 2012 and 2014.

Let's examine the similarities between MVC and MOVE. Both contain Models and Views. In MOVE, the Model contains the things that your app knows. This is true in MVC as well, but there may less methods or functions that "do things" in the Model of the MOVE architecture, like store a username in the database from the User Model, for example (because of Operations). The Views are essentially the same. They would contain the user interface logic.

A major difference of the two architectures is the concept of Operations. The Controller from MVC is gone and replaced by Operations and Events. Operations are everything your application does. Operations are generated by Events (like clicks, changes, navigation, etc.). In MOVE, event-operation chains will be scattered throughout your application's code, and you won't have a way of managing them all from a single file or directory (Century Link, 2013).

I think MVC is the better choice for the Little Library application. It seems as if MOVE is just an extension of MVC where the Model is simplified by unnecessarily separating Model logic into Events and Operations. Events generally come from a source, they are triggered by the Model or the View. Why not just keep them in place?

Century Link (2013, August 4th). Putting the MOVE framework in proper perspective. Retrieved from <https://www.ctl.io/developers/blog/post/putting-the-move-framework-in-proper-perspective>

\_\_\_\_\_\_\_\_\_

Reply 1

Hannah’s explanation of the differences between MVC and MVP is easy to understand, even if you have never heard of the Model-View-Presenter design. It’s simplified, “With MVP, the view creates the presenter and creates events the presenter responds to”. I agree that the implementations are similar (MVC vs MVP) with slight variations. The MOVE architecture also accommodates an alternative way to handle user interaction. Model and View seem to be present in most designs, but there are opinionated solutions for how to handle things happening within a given application (controller, view-model, operations-events, etc.).

I think that new design patterns will continue to emerge as technology advances and new web frameworks are developed. These will reflect the preferences of software developers as well as embrace performance improvements related to hardware.