Design Patterns - Choices and Implementation

We are using several design patterns for this project. In the behavioral view, we are using the State pattern since the overall progress of the program is fairly simple, and can be modeled as a series of states, state changes, and state change conditions. This is demonstrated by the use of an activity diagram in the behavioral view. Most of the components in this view can only be reached after certain conditions in a previous component have been met. So the components here are the states of the program, the conditionals are the state change conditions, and the edges are the state changes that result from those conditions. In the structural view, we are using the Composite and Bridge patterns. We are using the Bridge pattern because the components of our project can naturally be divided into two areas: a user interface, displaying what needs to be seen by the user, and the underlying logic which determines how different events change the interface. The event bus in our structural view (which is an event-driven architecture) is the "bridge" through which the UI and internal logic interact. In reality, this will just be a series of different events, rather than a dedicated component or program itself, but this is still a useful abstraction. We are using the Composite pattern because there are many extensions on the GamePiece class (one for each kind of piece and one for a "null" piece, or empty tile). So the extensions of the game piece class can be modeled in a small tree-like structure.