# Warzone Architecture Design



Figure 1 - Warzone Architecture Diagram

Figure 1, shows a package diagram giving a high-level view of the Warzone application and how each package maps into the MVC (Model View Controller) architecture of the system. Details of each package is provided below.

Note: packages in the diagram contain some example classes that form a subset of the overall set of classes that will eventually be realized by the end of build 1.

## Model Package/Module

* Maintains and manages the state of the application, i.e. game functions / entities such as Map, Player, Order, Country, etc…
* State change requests are defined by I\*Model based interfaces, where \* is replaced by the model defined class names. State change requests are initiated from the controller.
* State query requests are defined by I\*ModelView or I\*Model based interfaces, where \* is replaced by the model defined class names. State query requests are mainly initiated from the view based classes, but may also be requested by the controllers.
* Selective classes will make themselves observable, enabling other classes (mainly view based classes, but not limited to), to be notified whenever changes are made to the underlying game state (ie. Observer design pattern).

## View

* The game view will interact with the user using the system console.
* Renders the state of the application taken from the Model.
* Processes user input as gestures to the Controller. Note that the controller may also invoke methods in the views for user input.
* Accepts state change notifications from the Model, using a push model (Observer) which may trigger the view to re-render the output.
* View classes implement an I\*View interface (replace \* with the name of the view class).
* Controller classes dictate which views are presented to the user at different points (ie phases) of the game (e.g. map editor, game startup, game play).

## Controller

* Holds the behavioral logic of the application
* Selects/creates the views and maps user gestures into actions and state changes in the Model. The controller may also invoke methods in the views to accept user input when needed.

## Common

* Holds common classes used by the other packages in the system
* Does not reference any of the other packages in the system (Model, View, Controller) avoiding a possible circular package dependency