DESIGN DECISIONS

There were several design / architectural patterns integrated when creating the design for the Neureset Simulation program, and these patterns were subsequently employed in the implementation.

ARCHITECTURAL PATTERNS.

The principal architectural pattern used was Model-View-Controller, as explained in code documentation. The *main-window* class served as the controller class, and it was the orchestrator between the *model* class, which served as the model in MVC, and Qt designer's UI which serves as the view.

It handled GUI events, told the model to update itself, and then re-rendered the view with the updated model state.

DESIGN PATTERNS

1. ***The Observer design pattern.***

The controller class, main-window, observed the model class, the subject, for any changes to its state. The model class emitted a state changes signal which the model class then received and updated the view accordingly.

1. ***The Singleton design pattern.***

The model class followed the singleton design pattern. Exactly one instance of this class was and could be created. This allowed for easy, direct access from any model-related business logic classes to the model class. This was essential so that these business logic classes could add events to the model's running event loop with ease. It also made sense to have a singleton model class as only one Model instance was allowed for this Neureset simulation.

OTHER DESIGN DECISIONS

The other main, more general design decision was to have the model integrate a running event loop. This was crucial for having an easy way to trigger events such as pause timeouts during a session which would trigger in the future but that could easily be canceled in the code if necessary, such as when session progress resumed.