Design Document
CMPT 370

Group C4
Jack Huang
Brandon Jamieson
Ixabat Lahiji
Daniel Morris
Kevin Noonan

Architectural Choice

Our team's design makes use of the model-view-controller architecture. We chose this architecture because it plays to our team's strengths, as we are all most familiar and experienced with it. One benefit of this is that model-view-controller cleanly separates two major components, the model and view. This allows us to develop the two separately without relying on each other, as well as providing the user with a clear, bounded interface to the system. On top of this, the architecture keeps all robot-handling confined to a single major component: the model.

The view will be event-driven, handling interactions with users and relaying information to and from the controller. It will serve as an interface into the system and will inform the controller of user-input. The controller will also be event-driven, handling interactions with actors by manipulating the model. It will receive and handle user-input from the view, as well as updating and fetching from the model as needed. The model itself will have aspects of both blackboard and database styles due to the components inside. The robot librarian will serve as a database to be accessed by the controller for operations involving the access of robot teams. The game's board will act as the blackboard, with the controller manipulating the state of it as gameplay progresses.

One architectural option we considered instead of model-view-controller was a modified three-layer architecture. The top-layer would have been event-driven and handled GUI interaction. The middle-layer would have been blackboard style and the bottom-layer database style. Intermediate information would be stored in the middle-layer for game use, as well as receive data from the bottom-layer. We reasoned against this architecture because it did not encapsulate the guts of the game to one major component and instead, divided it among two of the layers. Model-view-controller provided a clear division between the guts and the rest of the system so we agreed upon it as our overall architecture.

\/ UML STUFF GOES BELOW \/