Architectural Design Description

The architecture used to create Bug Tracker is based on the B-C-E Model View Controller Design pattern.

Boundary classes are all the graphical user interfaces incorporated into the software, each boundary class identifies a use case. As the application is database driven, with each interface continually drawing and sending data, just one database controller class is used to handle this functionality.

The database controller class interacts with the boundary classes to provide access to the database. This controller class also interacts with the User and Bug entity classes, creating and returning new bug and user objects. The boundary and entity classes do not interact at all, as the main business logic is contained in the controller class. Business logic for this application is predominantly SQL statements with minor use case specific functionality such as checking password length, storing reputation points, etc. This is why only one controller class is used to keep the design simple.

The above diagram displays the architecture from top down with the log-in or Register screen being the point of entry. A user can also enter as a guest to view the main window which contains the search bug facility. Guest users can view the main window, search bug and search user interfaces.

Once a user is logged in they gain access to sub-systems via a menu on the main window. The user object created on log-in is passed to each sub-system from the main window and checked for access rights against its type. Eg Reporter menu will only display sub-system items a reporter can use.

In summary, user interaction is via the boundary classes in the form of a graphical user interface. Boundary classes interact with the controller to request and accept data as the controller contains the main business logic. Entity classes contain all functionality for the creation of a User or Bug, eg setters and getter and interact only with the controller class.