**Architectural Improvement Summary**

**“BetaGo”**

**Proposal for Architectural Change**

After reflecting on the current source code and architectural design for BetaGo, it appears that the program could benefit from improving and redesigning the loose MVC pattern that has already been attempted. Since BetaGo seems to be a project that a pair of students worked on in their leisure time, architecture may not have been a concept that was being considered during the developmental process, it is common students create games such as BetaGo in order to test their programming skills, rather than develop on higher level concepts such as architecture.

The current architectural issue is that there is a limited separation of concerns in the system because of the View and Controller aspects of the MVC being compressed into the Main class of BetaGo. Main consists of roughly 600 lines of code and has 10 nested classes, which are all responsible for different aspects of the View and Controller. Having this many nested classes seems a little unnecessary since they don’t seem to necessarily **need** to be nested classes. Therefore, I propose that an architectural change that would benefit the system would be to extract the nested classes from within the main, and to separate the View from the Controller in order to promote separation of concerns.

By implementing this change, I would be dividing out the BetaGo’s tasks more evenly so that each component in the system would be responsible for only a small number of things, rather than acting as a God Class and having an overwhelming number of responsibilities. This would achieve the desire of having overall higher cohesion in the system, which is regarded as being important in the overall design of a program. Following the MVC pattern is also considered to be good architecture because it de-tangles what could otherwise be “spaghetti code” that is hard to read and make sense of and splits it into three distinct components. Making this modification would be particularly beneficial to the programmers of BetaGo because it makes the code a lot easier to digest and understand, but also to build upon in future which is important because the project is currently active and is still being developed.

The modifications that I am proposing will bear no harm to the existing functionalities of BetaGo, and will simply enhance its readability and extensibility to improve the overall architecture of the code.

**Notes:**

Once I removed all the nested classes out of main (without fixing any syntax at all or changing any links), the only errors were occurring in the new classes themselves and in main (none in the model).