*3813ICT – Software Frameworks*

*Milestone 3 - Minesweeper*

Student: Zak Barker

Student#: S5085150

Subject: 3815ICT Software Engineering

Email: Zak.Barker@Griffithuni.edu.au

* 1. ***Design Principals***

Least privilege is a software design principal which relates to access rights, as does a fail-safe default. Least privilege sees that any entity should only be granted the least possible privilege necessary for that entity to complete its job. Fail-safe defaults is the idea that an entity must be given explicit access to an object, otherwise it is denied access to this object. The two work in conjunction as a method of preventing unnecessary access to promote the integrity of the system as a whole. Encapsulation is the process of encapsulating separate components of a system so that they are independent of one another. This leads into the ideas of coupling and cohesion. Coupling is the interdependency between two components and this should be minimized as a failure of one component should ideally - not affect the functionality of another. Cohesion describes the effective relationship between operations within a specific module. These concepts encourage information hiding. This is the concept of keeping the internal functions of an interface invisible to an entity that does not require the knowledge. The entity should simply know how to operate the interface as opposed to knowing how it works internally.

* 1. ***Design Process***

The first step in the design process for the Minesweeper implementation was prototyping. This involved created an initial version of the game to facilitate tests, development and functionality. Prototyping is a critical aspect of the design process and a working prototype was the first goal. Initially, Models were created based on conceptual designs for minesweeper, however, over the course of implementation, these models became unsatisfactory for functionality. The process became more of an agile approach in which pieces were modelled, built, tested and then integrated into the main project. This was a small scale implementation but the principals were essentially agile and a new overall model resulted from this along with an improvement in efficiency. The end result was an overall idea for software design and architectural designs. This process also helped with the process of identifying which pieces could be decoupled and aided in deciding which functions should be related to which functions and which functions could be factored out and re-used.

* 1. ***System Models***

Ahead is an image of the original class model used for the creation of this project:

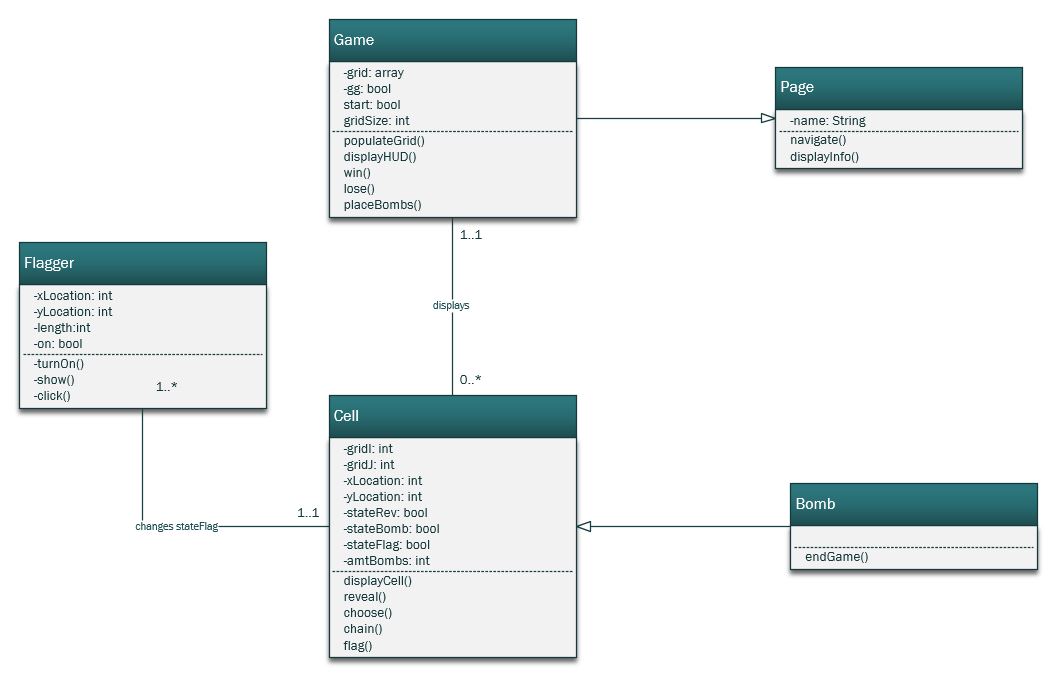


Figure 1 – Original Class Diagram

This was revised throughout the software design process in order to facilitate cohesion and reduce coupling. The design has become slightly more complex but allowed for separation of responsibilities across various aspects of the implementation. The new design is as follows

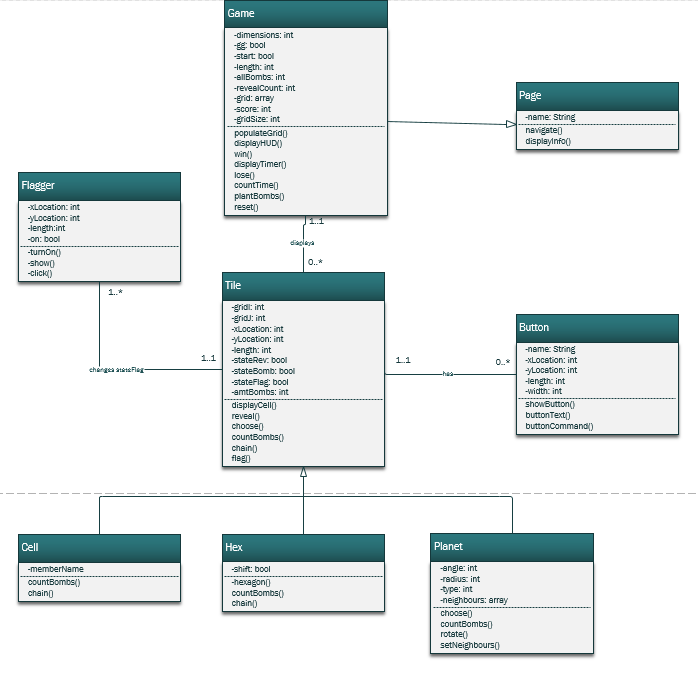


Figure 1 – Revised Class Diagram

The tile types were separated into their respective subclasses of the Tile. The common methods and members were factored out so that the subsequent subclasses would be able to hold information crucial to that type of tile. For instance, in a hex game, each tile row needs to know if it is odd or even. Furthermore, slightly different algorithms were necessary for the same function in different tile types. In this case, the original can be overridden. This adds to a limiting of coupling between tiles and the gameState. This design is much more efficient than the original class structure. The Game class has been extended to include more functionality. A button class has been introduced to disconnect the Game class from the navigational aspects of the software. This is now handled separate from the game itself. Finally, the Game itself has been modified to hold all of the rules for the game in separate functions along with all of the information required for a game state. Each method pertains to a rule within the game and the members facilitate this.

* 1. ***Design Paradigm***
  2. ***Software Architecture***
  3. ***Software Programs***
  4. ***User Interface***
  5. ***Model-View Controller***
  6. ***Good Software Design***
  7. ***Design Documentation***