1 Team Structure & Development Process

1.1 Team Structure

In order to develop a structured approach to task allocation, the software engineering tasks require were structured using the Administrative Programming Team [?]. This consists of the following roles:

- **Project Manager**
- **Librarian**
- **Configuration Manager**
- **Toolsmith**
- **Quality Assuror**

It should be emphasised that each person was not solely responsible for the tasks associated with their role; their responsibility is to coordinate these tasks within the team by proposing, implementing and maintaining effective procedures to acheive this.

In tandem with this, the team was further divided into two subteams for development:

- **Modelling and GUI Team:** responsible for development of any 3D models required including the final model of the ship. In the later stages of the project this team developed the graphical user interface.
- **Core Implementation Team:** responsible for all other development tasks, including implementation of the navigation mesh, population model, etc.

By seperating the development of the user interface from the development of the underlying program logic, the team aimed to promote a Model-View-Controller design. [?, Ch 6.3.1].

1.2 Development Process

One of the challenges of designing an evacuation simulator is defining a level of accuracy which can be deemed acceptable with respect to the project's resources, and then translating this into an effective design which balances the use of up to date techniques with an implementation plan. Many of the techniques that this project aimed to implement are complex. These techniques are discussed further in the Research section.

Ultimately, it became clear that the most tangible way to make progress was to use a strategy of incremental prototyping [?, Ch 2.3.2]. This would allow the team to progressively 'scale up' ideas and to investigate the feasability of implementing certain principles in a structured manner. However incremental prototyping carries considerable risk which must be addressed:

- A tendency to produce low quality and difficult to maintain code.
- Difficulties in managing change.
- Tendency to sacrifice quality assurance and documentation because of poorly understood aims.

To mitigate these risks, several techniques taken from the field of agile development [**?**, Ch. 3] were employed as follows:

- **Division Into Subteams:** The team was divided as outlined above so as to allow these subteams to work in parallel on orthogonal tasks, This reduced communication overhead and the difficulty of managing change to the system.
- **Constant Refactoring:** Before the completion of each prototype or upon fixing a defect, significant reactoring was undertaken to improve code quality.
- **Pair Programming:** This technique was particularly helpful when fixing defects related to navigation (see Implementation) due to the complexity of these defects.
- **Test First Development:** wherever the understanding of requirements was sufficient to allow it, test cases were developed for a feature before they were implemented. The full testing procedure is discussed in the Evaluation section.