Hello and welcome to this presentation on Robustness and Sequence Diagrams, I, James Moran, will be giving this presentation on their use in the Game Café, let’s begin.

What are Robustness and Sequence Diagrams? For Robustness Diagrams, they are a graphical way to depict use cases, that is also a ‘sanity check’ on Use-Cases (to make sure the Use-Cases match-up to what functionality the User would want from the system), that also allows the team to uncover new classes, which would not have been previously identified for the system. Sequence Diagrams allow a team to design the system in detail, considering the methods/functions the classes in the system will have and the order that these methods/functions are executed.

In the Game Café, the Robustness Diagram is as follows: Considering Staff Members of the Game Café, they would want to manage information of Members, Bookings, eSports Events, Hardware and Software (an example of a boundary), after manging this information, check to see if it is correct and that this information is not already present in the database (examples of controls, is not correct, show an Error Dialog (another Boundary)), before adding it to the Information Database (an example of an Entity). A similar procedure is taken for if there is a ticket available for an eSports Event (with similar Boundaries, Controls and Entities).

Moving on to the Sequence Diagram, this details the ‘timeline’ for the Use-Case of a Game Café Staff Member adding information to the system’s database. They start off by finding the target category of information, that they would wish to add, then the system requests (via the Windows Form Interface) details to be entered by the User, whom enters these details, which are validated for correct format by the system, as well as checking against the database for duplicate values, if the information is of a valid format and there are no duplicates in the database, then this information will be added to the database.

Advantages and disadvantages of the MVC Design-Pattern, are detailed here, with the Advantages being that of Supporting Multiple Views (allowing the user to change the appearance of the system’s forms, as the view is separated from the Model) and Accommodating for Change (most notably, that of changing User-Interface Requirements, for if the User wants to view the application on another device). The potential disadvantages are that of Complexity (introducing new levels of indirection to the project, as well as taking into account the Event-Driven systems for User-Interface code, which can become more difficult to debug) and the Cost of Frequent Updates (even with decoupling between the Model and the View, as the developers of the Model should at least consider the View, when making changes to the system Model).

This slide details the references used in this presentation, thank-you for taking the time to watch this presentation and goodbye for now.