Layered View Rationale:

This view was defined to aid all the stakeholders in question as it provides a very high-level view of the system implementation. Maintainers and developers will be able to extract the general control flow of the program, making development or debugging simpler. Each layer has a certain functionality that no other layers should have, making it clear where a certain module should be implemented.

Module Decomposition Rationale:

This view promotes modifiability and maintainability. By decomposing the main modules in smaller modules, we have smaller more concise units of implementation, this helps to lower coupling between classes therefore boosting modifiability. It helps futures developers and maintainers to reassess the way the modules were implemented helping them to either integrate new functorialities the system may need, or to correct already existing bugs.

Module Uses Rationale:

This view serves as an aid mainly to developers, but also to code maintainers and project managers. It shows a more in-depth view of the modules implemented and the dependencies between each other. These dependencies are to be considered in future iterations of system implementation or for debugging purposes. By seeing which modules depend on which, we can see clearly the general control flow of the system, increasing the testability.

Web Client C&C Rationale:

This view is useful for both implementation and deployment purposes. The end user can access this view to see what is accessed when using the web application, giving some more in depth information about the inner workings of the system. This may increase system usability as the user acquires more information. It also boosts system maintainability, by seeing which components interact with each other and how they do it during runtime eases the testing and debugging process. For instance, if the screen is displaying wrong date regarding the upcoming matches of a certain player, then we know that either the Action component is not processing the input data correctly, or the session beans that conduct business logic are not validating the data correctly before retrieving information from the database.

This view also boosts modifiability, the developer can extract from this view information regarding the components and connections to access the web application to, for example, register a player into a tournament, and may want to add additional functionalities to that use case, or add a new use case all together following the flow control of the components.