# Block Diagrams Description

In our architecture, the Home Audio System application presents a Model View Controller(MVC) architectural pattern. The MVC concept is beneficial since it gives us the possibility of separating and managing the application into three different modules: the Model, the View and the Controller. The View components of the MVC represents the presentation layer while the Model and the Controller are both considered into the business logic. Thus, it is easier to configure one module, the presentation for example, without having to modify the other ones, the business logic in this case. The maintenance of the code also gets simpler and it is more transferable. Concerning the transferability, since the architectural design allows us to change precise modules of the application, it is easier to transfer the application to different platform (Desktop, Web, Mobile) as the major change between the platforms is only the presentation layer. As a matter of cause, It is the reason why the same MVC architectural pattern is used on the three different platform.

StartMenu Application Frame

This represents the start of the programs. After the user starts the StartMenu, it launches the Home Audio System application. Since the StartMenu is only the initial view, it doesn't contain or share any data. However it is the necessary pipeline to access the MainMenu: the Home Audio System Application.

Home Audio System Application Frame

This is considered the MainMenu of the program. It displays the music, location and volume management as well as the streaming of the music. While it displays all the available actions, it does not perform any modification on them. Once an event occurs, this subsystem relays the information to the controller which deal with the information accordingly. For example, if the user wanted to access to a different view of the system, it is the controller that will update the Home Audio System Frame to its new view. Also, the Home Audio Application Frame subsystem as a deep relation with the HAS model subsystem, as they continuously share information to keep track of the updates and make changes to the View if necessary. In fact, all the data displayed in this Frame is to be founded in the HAS Library Model.

Home Audio System Controller

The Home Audio System Controller connects all of the models together, and displays it through the view. If any change is made in the view, the controller subsystem is in charge of update the models accordingly. Hence, the controller subsystem shares data with both the Home Audio System Library Model and the Home Audio System Application Frame. In a conventional MVC architectural pattern it is natural to have more than one controller to handle the different models (Play music, Change location, Manage the songs), however having one controller allows for high-cohesion and low coupling, as it is not necessary for all of the models to be coupled to one-another. Also, the HAS Controller is the subsystem which manipulate the persistence of the system. It saves data to the persistence and also loads data from it to display it on the View.

Home Audio System Persistence

This subsystem provides access to the persistence layer. The persistence layer is the database of the system where the information is organized to be easily accessed and managed in the current session and for the upcoming ones. The Home Audio System Persistence is accessible by the HAS Controller to save new data or load an existing one to display on the HAS View.

Home Audio System Library

The Home Audio System Library represents the models of the system. It is the core of the application. Every objects in the application has its own structure and, both the structure and the relation between the objects are described in the HAS Library. Hence, it describes the behavior of the application as it contains the logic and the rules of the entire application. The Model which is occasionally changed by the controller is then sent to the HAS Application Frame to update the View. In other words, the Model actively shares data with the View and receives data from the Controller. However, it doesn't send any data to the Controller as the user doesn't have direct access to the HAS Library Model.