**COMP 413- Loyola University Chicago – Project 4**

**Name:** Alessandra Velozo **Name:** Igor Schneider

**Domain Model**

**Controller**

**Trait Controller** - extends activity, updates the dotView and has a dotModel and a squareModel

**DotGenerator, MonsterChanger, MonsterMover -** extends AsyncTask and calls methods in the Dots class.

**TrackingTouchListener** - extends View.OnTouchListener and calls methods in Dots class.

**Model**

**Dots class-** has a ListBuffer of dots and squares . It also has a DotsChangeListener

**Dots Object –** has a DotsChangeListener

**Dot -** has a square and saves its color and diameter

**Squares-** has a ListBuffer of squares

**Square -** saves its position ,side and if its full or not.

**View**

**DotView** – Contains the android graphics classes and a Dots variable

**Report - Development Decisions**

During the design of our application we had to discuss and decide the best design to develop the software. Consequently, we decided that our application would be designed based on the Model View Controller (MVC) pattern.

We chose this design pattern in order to separate application's concerns, consequently the Model has the business logic, the View manages the user interface and the Controller is responsible for the navigation & input. Using this approach we turned the code more simple and allowed code reuse. We decided to keep this design instead of Model View Adapter because in this approach the Adapter tends to take on a lot of the responsibility and gets more complicated. In our application has a trait called Controller and it is responsible for getting events from the user, informing the model and/or the view to change conformably. There are also there classes responsible for creating, changing and moving the dots and another to get the events. In the model we have the class Dots which represents a list of dots and the Squares that is a list of squares. In the View we have the DotView that creates the view for the user.

In order to create a good software architecture we maintained responsibility-based separation among the different building blocks, then we concentrated in each class its own responsibilities. Also, if we want to change the Dot behavior we do not need to change its class. Since our application is separated in different blocks, it turned the development easier and also made the testability possible. Concerning the concurrency, we had to deal with the multi-threaded programming by using the AsyncTask and creating three classes (DotGenerator, MonsterChanger and MonsterMover) that extends the AsyncTask. Each of these classes have its own Thread, which specifies different delay times.

We believe that designing this application made us have a better understanding on the design patterns learned so far and also comprehend the multi-threaded programming.