Data visualization customers are increasingly expecting web-accessible data visualization solutions. Unfortunately, browser technologies are still unable to cope with complex data analytics that require powerful algorithms, dense and intricate graphics, and that can handle large amounts of data.

There is currently a framework in place, called Perspectives, that provides the means to accomplish more efficient data visualization but has been developed as a desktop application.

The main focus of this project was to transport this functionality to the web while keeping the Perspectives as the backbone. The new system will rely on AJAX and JavaScript technologies to relay interaction events between the server and the client to continuously return rendered images as the visualization in a viewer.

My contribution in the project focuses mostly on everything that has to do with the visualization properties. There are many types of visualizations and each one has its own set of properties that can be dynamically changed by the user. My job was to set up the system in a way in which the correct properties are created when a user launches a viewer as well as relay the information between Perspectives and the client when a certain property is changed.

The current system consists of a Java-based desktop application used for data visualization. It utilizes the Perspectives framework, developed by Dr. Radu Jianu at FIU. Users are able to upload data sets and choose from a given set of viewers compatible with the uploaded data set type. The system then presents the user with an interactive interface, called viewers.

Each Viewer is constructed around the notion of Properties. Users can control the properties and the visualizers respond to those changes by altering its appearance and performing computations.

The general requirements for the system is to allow for similar functionality that is available in the desktop version of perspectives, but made available through the web, handling most of the processing on the server and having the viewers only show images being sent continuously.

The specific requirements that relate to my role are as follows:

- Create JavaScript translations of the different property types available in the desktop version.
- Display the necessary properties for each viewer type on the moment of creation.
- Communicate property changes made by the user to perspectives.
- Update property values, or the properties themselves (remove or add) at runtime if Perspectives signals a change for that viewer.*

*It is possible for viewers to be connected, in which case it may be the case that a change in one will modify another.

The online version of the Perspectives system was developed using a 3-Tier approach as shown below.

This approach was chosen because it was necessary to have a Presentation layer to represent multiple pages, along with client-side logic.

The business logic layer translates the property information sent from either direction.

Finally, the Data Layer is the Perspectives Framework that, among other things, provides the logic to render the multiple images, as well as keep track of the multiple viewers and their properties.

The main subsystem I was focused on was the Viewers Subsystem, shown below.

The Viewer class in the middle is where the call to launch/create a viewer is made. All of the clients-side logic necessary to create properties is found on the Ajax.js utility shown on the left. On the right is the Controller that communicates between the Viewer class and the Perspectives framework.

The system was implemented using Java, JavaScript, HTML and AJAX technologies.

Below is an example of a property function that creates an Integer Box:

```java
require(['App'], function(App) {

    var p = new App.PropertyBox();
    p.property = 'anything';
    p.value = 10;
    p.bind('value', function(value) {
        // change color
        // change stroke
        // change width
    });
}
```

Here is a snippet of a large method called addProperties that decides which properties need to be created depending on the viewer selected.

```javascript
addProperties = function(viewer) {
    // check if Viewer is valid
    if (viewer === undefined) {
        return;
    }

    // add and set property
    function addProperty(property, value) {
        // if property not already created, create
        if (viewer.getProperty(property) === undefined) {
            viewer.addProperty(property, value);
        }
    }

    // add and set properties
    addProperty('horizontalOffset', 0);
    addProperty('verticalOffset', 0);
    addProperty('strokeColor', 'blue');
    addProperty('strokeWidth', 1);
    addProperty('strokeStyle', 'solid');

    // add and set property from view
    function addPropertyFromView(view) {
        // check if view is valid
        if (view === undefined) {
            return;
        }

        // get property from view
        var property = view.getProperty('propertyName');

        // add property
        addProperty(property, value);
    }

    // add and set property from view
    addPropertyFromView(viewer.selectedView);
}
```

The new system that was created extends the Perspectives framework, developed as desktop application, by making it available online, while still having the interactive visualizations along with all of its functionality.

It relies on simple technologies available in most modern browsers in order to make it accessible to most people.

My focus was to make the necessary properties available on the web pages as well as to develop the property controls for the viewers that house the images in order to communicate this information between the server (where Perspectives is stored) and the client.

I am thankful for the help that I received from my group members Raul Tobo and Erik Franco as well for the assistance provided by Mershack Okoe.

**ACKNOWLEDGEMENT**