Architectural Design

High Level Architecture of Sensor Data Service Platform Application

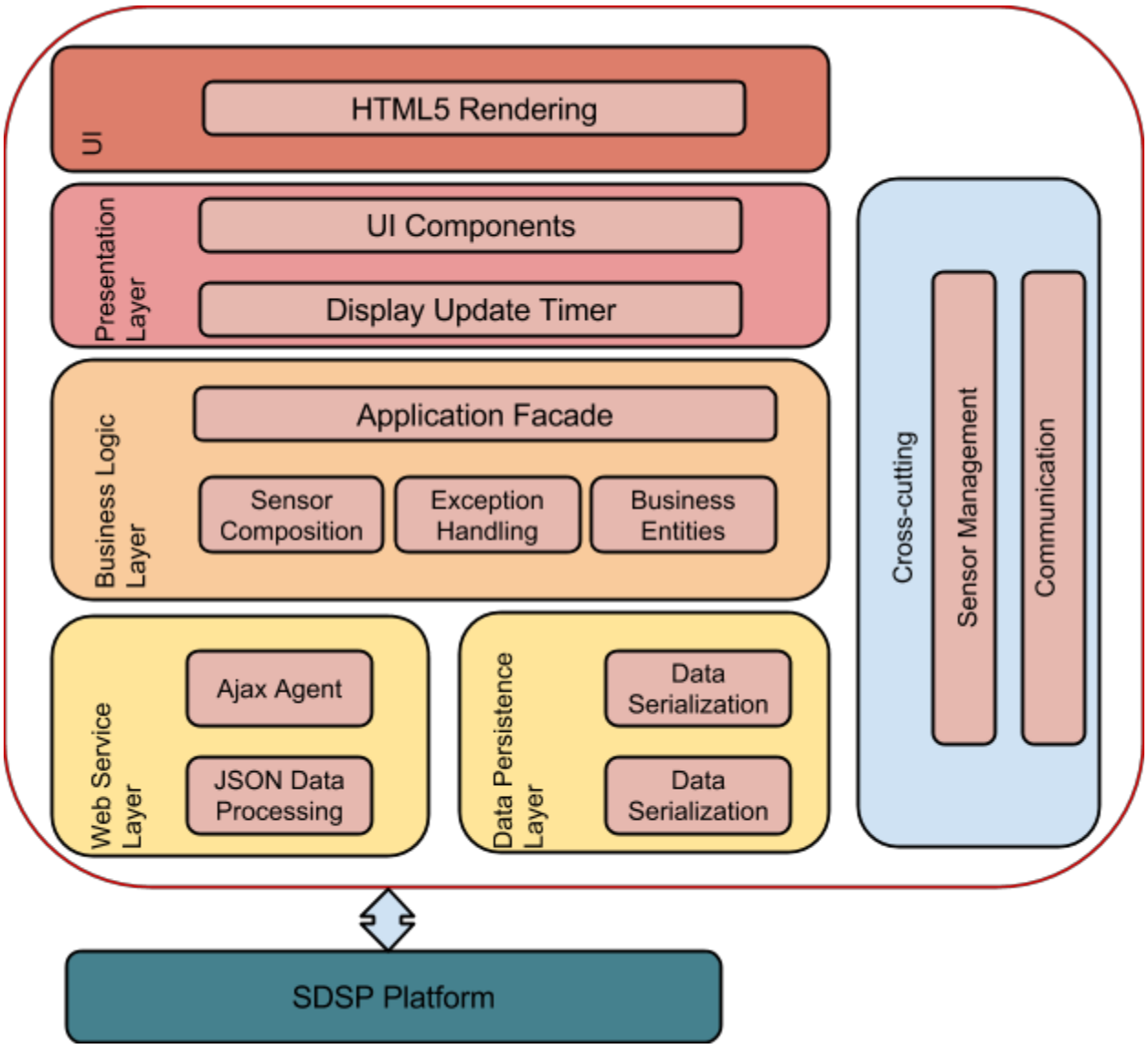

Figure: Virtual Sensor Design Tool Architecture

The above diagram (citation VirtualSensorEditor-Presentation-07262013.pptx) shows the overall architecture of the Virtual Sensor Editor Platform.

The figure shows the Sensor Data Service Provider Platform acting as the service layer to the Virtual Sensor Editor. From there, the various layers (Data Persistence, Web Service, Business Logic, Presentation Layer, User Interface (UI), and Cross-Cutting) contain key components,

features, and functionality.

As can be inferred from the project background, the team's major contributions fall within the Presentation, UI, and Business Logic Layers as exemplified in improving the sensor visualization dashboard, adding new components/control flows to the virtual sensor editor, exporting virtual sensor data serialization to XML, and implementing geofencing algorithms to support the future work of adding mobile devices as ad-hoc sensors.

System Functional View

The figure below (from Team Whippersnapper CMUSV Spring '13) depicts the system's functional elements, responsibilities, and interaction.

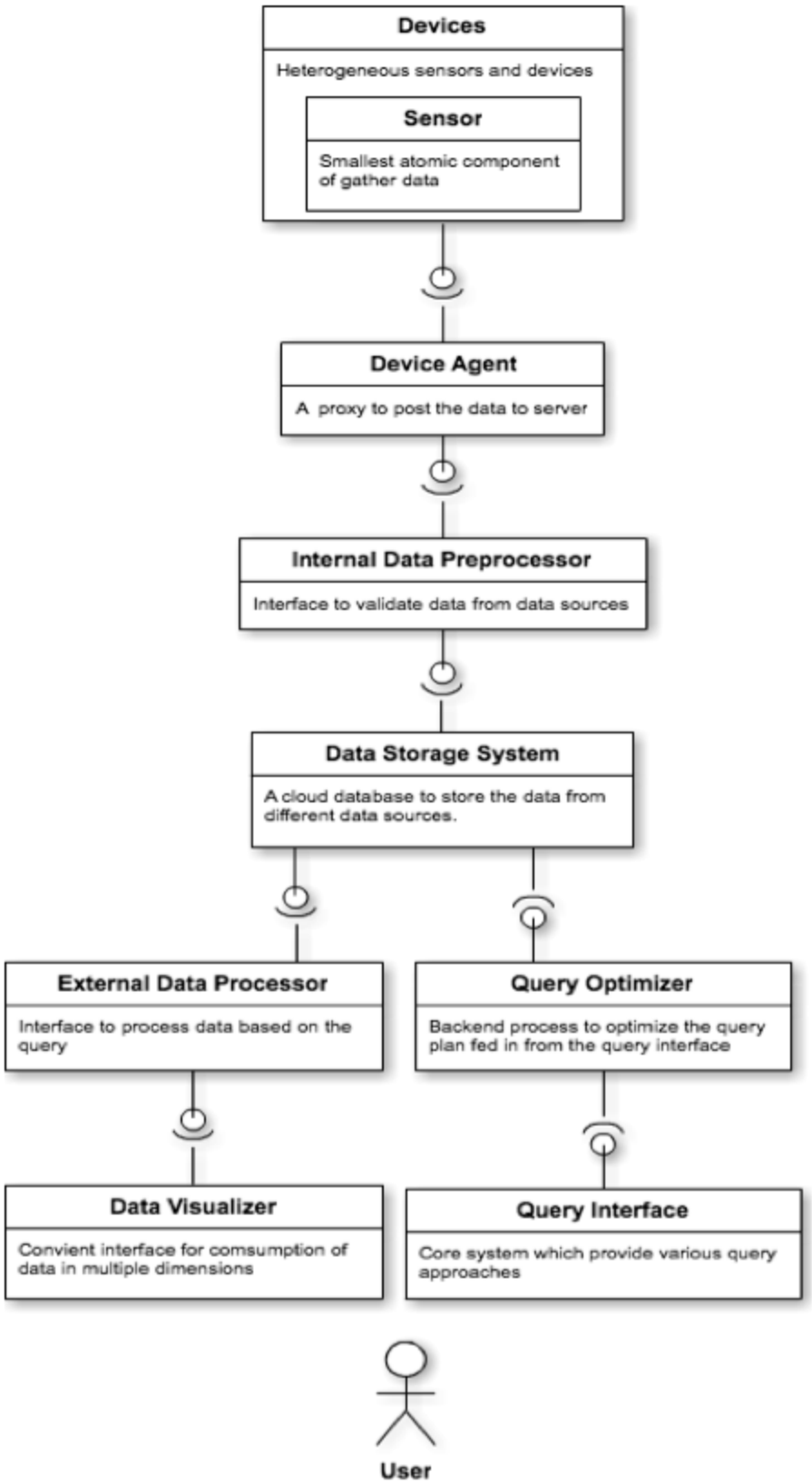

Figure: System Functional View

As seen from the figure, the devices and sensors collect data and get processed, stored, optimized, and finally visualized through the means of the virtual sensor editor for the end user. Team Facade's project and tasks focuses within the data visualizer area depicted on the figure.

Strategy Pattern: Virtual sensor serialization to XML

To facilitate the export of valid Extensible Markup Language (XML) to Taverna and Vistrails, both open source workflow management packages, the team utilized behavioral patterns which allow for the realization of common communication patterns between objects. In particular, the team made use of the strategy pattern. The benefit of the strategy pattern is that it separates out algorithms into classes which can then be plugged in at runtime. The figure below depicts the notion of a strategy pattern.

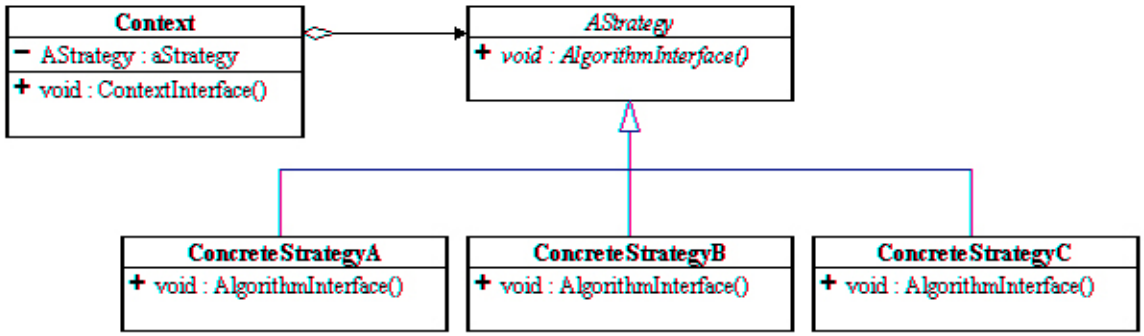

Figure: Strategy Pattern [citation: "Strategy Design Pattern." *DoFactory*. N.p., n.d. Web. 7 Dec. 2013. <http://www.dofactory.com/Patterns/PatternStrategy.aspx>.]

The team's task to convert the virtual sensor serialization to XML required many distinct behaviors where each behavior was implemented as a strategy. The serialization of the sensor data, originally expressed is JavaScript Object Notation (JSON) was first converted to XML using the x2js library with modifications needed to be made to the JSON structure. The next strategy involved an Extensible Stylesheet Language (XSL) transformation to a format that can be imported into Taverna. Finally, a command line tool (xsltproc) was used for applying Extensible Stylesheet Language Transform (XSLT) stylesheets to the XML which allowed for the successful import into the Taverna workflow.

The use of the strategy patterns allows the changing of an object's behavior dynamically without changing the object itself. A more detailed description of the team's implementation of this task using this pattern is found later in the report.

Design Pattern for Control Flow (Decision)

For the implementation of a control flow within the VSE application, the team made use of the observer design pattern. Because the team decided to implement a decision as the type of control flow, the inputs received from the virtual sensors directly corresponded to the output of

the decision box. This allowed the team to utilize the update method in receiving sensor information and using that information to toggle the outputs of the decision and have this be displayed in real-time on the virtual sensor editor canvas.

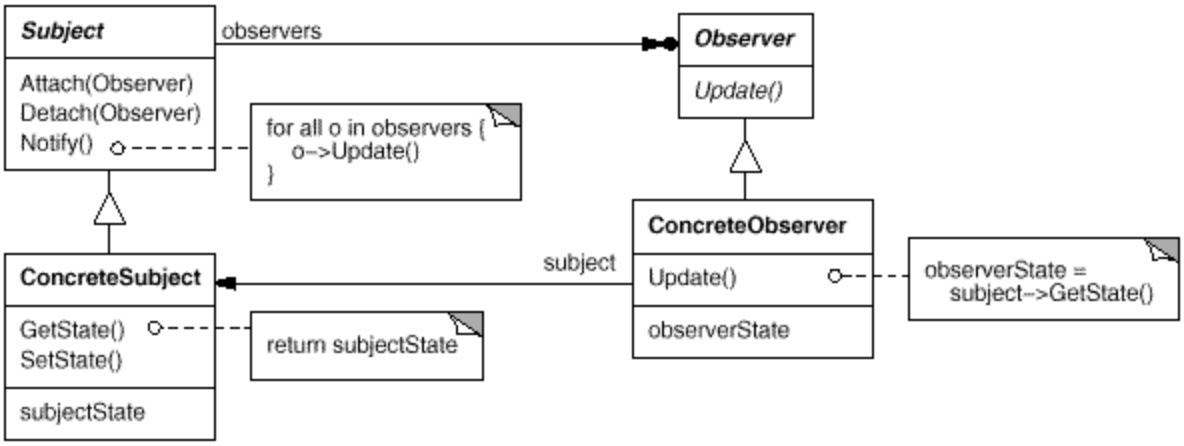

Figure: Observer Pattern [citation: "List of Design Patterns." *Design Patterns: Observer Pattern*. N.p., n.d. Web. 7 Dec. 2013. <http://www.bogotobogo.com/DesignPatterns/observer.php>.]