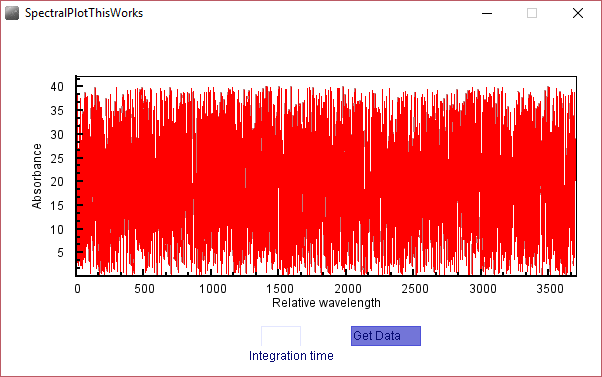
Design Summary

This document describes the design and architecture of a prototype charting tool that charts Absorbance VS Relative Wave length data obtained through the computer serial port.

# Goal

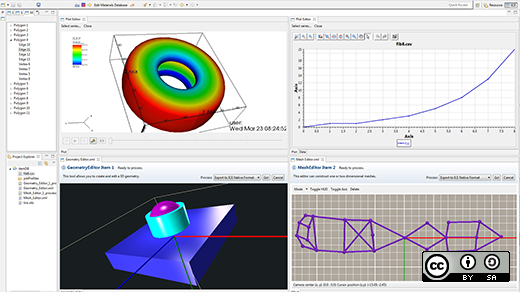
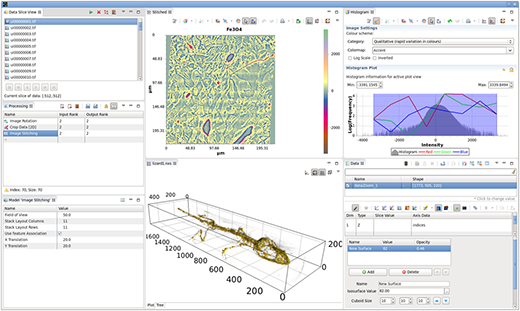
The following screen shows the original tool based on Processing 2.2.1 (processing.org).



The processing programming language is very well suited for rapid prototyping and data visualization, but when scalability and impact is needed it is not positioned well. The goal was to migrate this program to another platform, which will allow adding more functionality and productizing the tool.

# Platform

Eclipse (Eclipse.org) is a platform of choice of many scientific workbench projects. This article lists some projects based on it - <https://opensource.com/education/16/4/5-great-eclipse-scientific-workbenches>

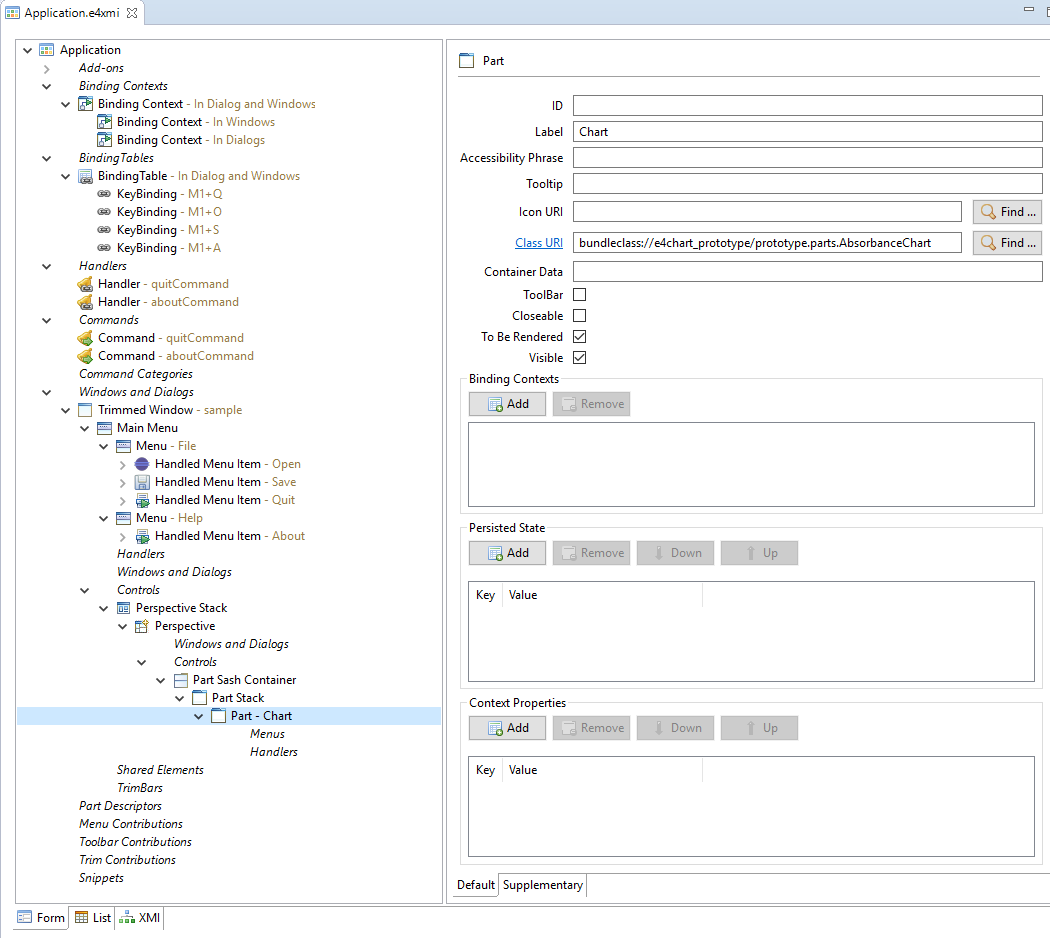


Few of the reasons to use Eclipse are:

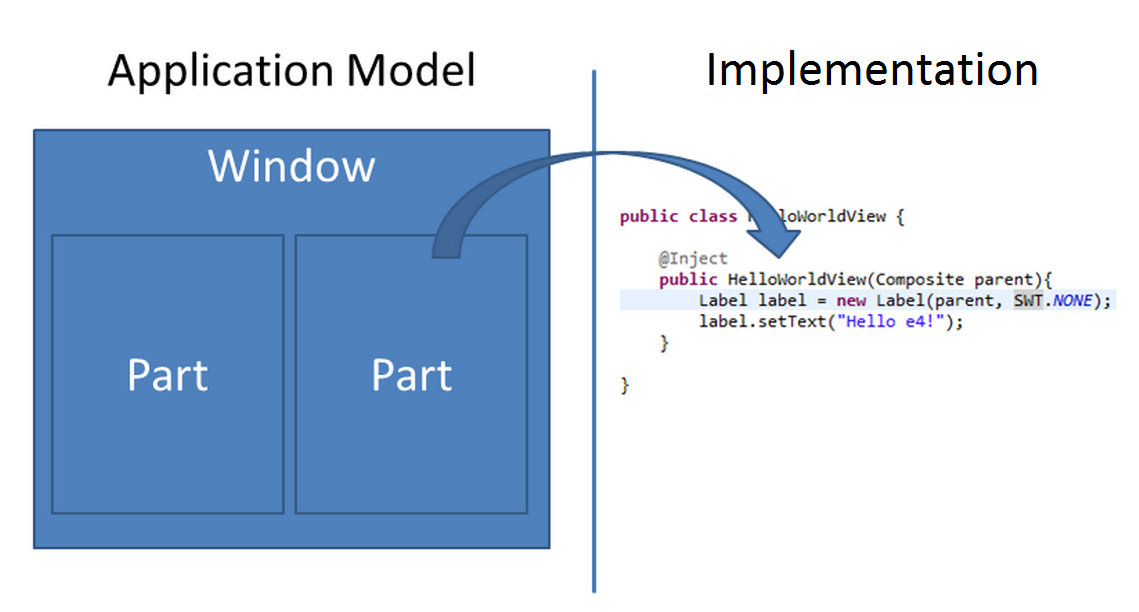
* Highly modular and extensible design. Based on OSGi model, Eclipse building units are called plug-ins. Each functionality is developed and deployed as a separate plugin.
* Open source model. Thousands of Eclipse plugins are available.
* Mature code base and large community of developers.

# Design

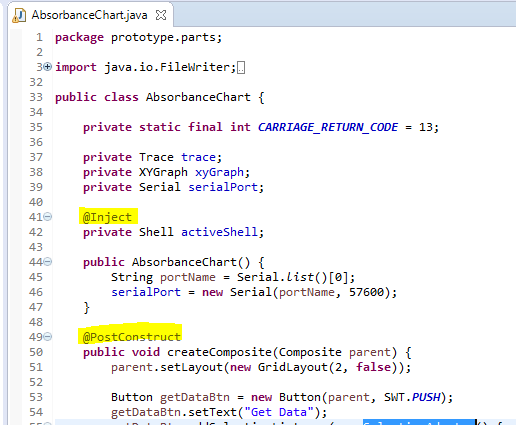
The central part of an Eclipse Rich Client application is the application model. It is usually contained in an XML file called “Application.e4xmi”. This file contains the main building blocks of the application. Eclipse provides a nice visual editor for this file. When opened it looks like this:



Many artifacts such as menus, keybindings, etc. are pre-populated by the Application creation wizard. The highlighted part is the chart that we implemented – AbsorbanceChart. Note how the XML file just defines a reference to the actual Java Class that will implement the user interface and the functionality. In Eclipse terms, a Part is an atomic UI panel (or view) and Perspective is a set of Parts arranged via certain Containers (Sash or Stack containers). The following image illustrates how the parts of an application model are connected later to their implementations:

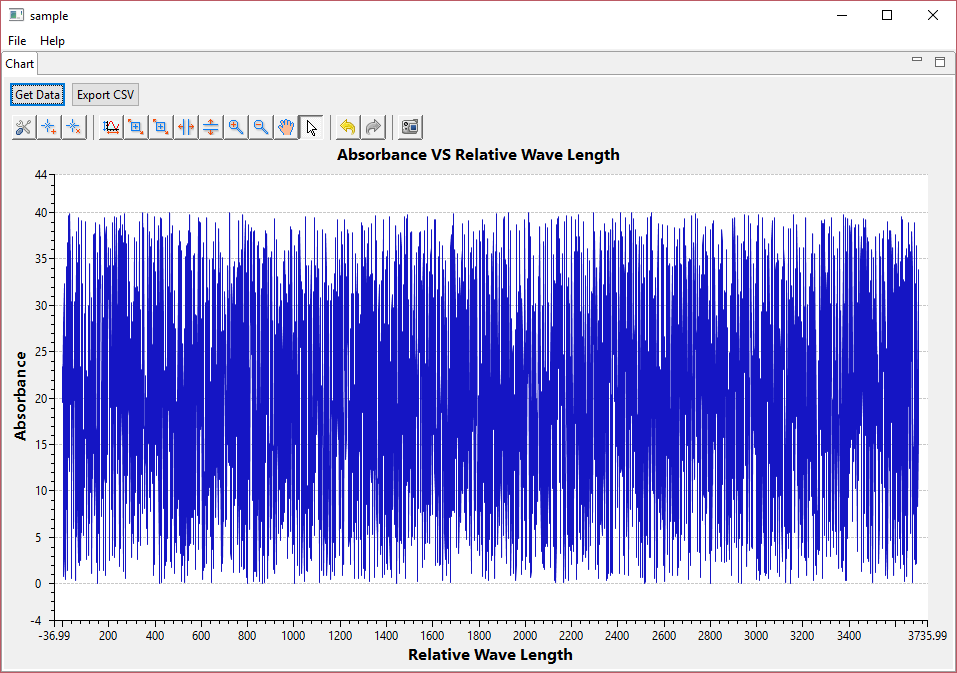


The main view of our application is implemented in the AbsorbanceChart:



Note that it does not extend a framework class, neither implements some Eclipse interface. It is a standalone Java Object that can easily be migrated to any UI framework. What Eclipse requires is adding certain annotations like @Inject and @PostContruct. The platform uses inversion of control principle (<https://en.wikipedia.org/wiki/Inversion_of_control>) to manage the context of the individual building blocks of the application. They don’t need to know about context, the platform will provide it instead.

The following screen shows how the Eclipse-based tool looks.



# Functions and components

## Obtaining the data

To obtain the data we use the same library used in the initial prototype based on processing. It is called Java Simple Serial Connector (<https://github.com/scream3r/java-simple-serial-connector>). In addition we wrote a wrapper class called Serial to utilize the library from our application. (the Serial class was actually migrated from the Processing code base to Java code)

## Plotting the data

To plot the data, we use the Nebula Visualization XY Graph widget (<https://eclipse.org/nebula/widgets/visualization/visualization.php>). This is a lightweight charting library for Eclipse, that allows quick programmatic creation of charts.

## Exporting the data

To export data to CSV we use the Open CSV Java component (<http://opencsv.sourceforge.net/>). The .JAR library is added to the list of libraries in the application.

# Development considerations / constraints

* We use Eclipse E4 development platform (as opposed to the older Eclipse 3.x style plugin development).
* We use 32bit Java 8 and also the Serial Connector is 32bit version. They need to have same 32/64 bit versions.
* We use Eclipse for RCP and RAP developers, Mars 1. Release:

