# Introduction

This is my attempt at testing 3 different technologies to do 3d interactive graphing for CaIntegrator2. The first approach is using Java/Swing/Java3d/Applet’s as a delivery method, based on the GenePattern PCA and TIGR MEV (Multi Experiment Viewer) code as a starting point. The next one is Adobe Flex and then JavaFX. Each section will briefly explain what I discovered, some of my personal opinions on them, and a very rough estimated LOE if we tried to implement them as well as some links of interest on the technologies that I used to base my decisions on.

# Using Current Methods at NCI (GenePattern PCA / TIGR MEV)

The easiest way to quickly get 3D graphing into Cai2 is to use the TIGR code (open source) and modify it similar to how the PCA viewer was modified. These both use the J3D jars, so the client will be required to have Java3d installed so an Applet can run. The PCA viewer allows for a 3d graph with rotation and point selection and the input is based on ODF files and a GCT file. The main thing is the PCAContent3D.java which is the content pane for the 3d graph, this was taken and modified from the TIGR Content3D.java, what I like about PCAContent3D’s implementation is that it uses a List of XYZData which can be created out of anything it just contains the data points to graph, we should do something similar so that the graphing mechanism is generic but the way to get the inputs isn’t (similar to our other graphing mechanism, such as Gene Expression plots).

## Some difficulties to overcome:

1. Writing Swing based application to run in an applet, the development team is currently not using this technology so a little learning curve or refresher will be in order.
2. Adding on to the current code to allow for a wide range of user operations including zooming, panning, rotating (currently rotating works), and multi-selection of points. Currently it allows you to select points one by one.
3. Centroid/ellipsoid around groups doesn’t exist on PCA, something would have to be hand written, and an algorithm would have to be created to figure out the ellipsoid properties.
4. Allowing the classes to be colored is a new feature as well.
5. Basically we can pull out a generic 3d graphing library with minimal functionality and build on top of that in SWING to create a webstart or embedded applet application.

## Some of the items that Gene Pattern's PCA viewer gives us out of the box that aren't in the TIGR MEV code are

1. A nice layout with the data points on the bottom in a table format
2. When hovering over a data point it shows a label with the information about that point.
3. When clicking a point it changes the color and jumps to that point in the table.
4. GenePatterns' version lets you select the 3 principal components you wish to graph and shows the genes from there, but the MEV lets you choose between Samples or Genes to use in the calculation.
5. One pitfall is that the points are always drawn as spheres, which can take a while to load if there's a lot of data, the MEV lets you select between spheres or points.
6. Neither lets you pan, zoom or select groups of points, and neither lets you color based on annotated values.

## Communication between Server and Applet/WebStart

The need to communicate from Applet to server is essential to be able to save gene lists or sample lists of interest. The easiest way to do this is through **object serialization** from Applet to Servlet. The basic communication between the two might look like this:

1. User goes into the “Analysis Tools” section, selects “Multi Experiment Viewer”, and is taken to an input page which allows the user to define the classifications (different Annotations to classify data by) and genomic data input (either all data or a subset defined by a query or subject list). Also, on the web form, the genomic data could also come from a saved Gene List or Sample List
2. Cai2 will then create the input MEV file based on this, and allow the user to click a link which goes to a JSP that has the embedded MEV applet startup.
3. While the user is doing stuff in the MEV applet, they might see some genes of interest or samples of interest, so we would want to lasso them (depending on which tool they’re in) and right click and do a “Save-As Gene List” or “Save-As Sample List”, this would allow the user to name and save the list
4. After it’s saved, the user can view and use that list just as they would if they saved it through the web interface.

Here is how it might look architecturally.

1. The JSP starting the Applet will need to pass the input file, since it’s a client / server architecture, the input file might have to be served up through a Servlet or some other secure means (not sure exactly).
   1. The Applet also needs to know the URL for the actions to call them, this can be given to the applet at startup time via the JSP calling it.
2. After a user chooses specific genes on the tool, it will need to serialize the list of strings (comma separated list of gene symbols) and push them to an action which takes them in and saves them for the given user.
   1. Another tricky piece here is having the http session stay up for the applet so that the workspace is still in scope and it saves this for the correct user.

## LOEs

1. Create caintegrator2-graphing-jar package which will produce the .jar artifact that has gene pattern’s PCA viewer in it as well as a generic 3d graphing viewer (to be written later). Also integrate GenePattern’s PCA viewer as-is with the ability to use it directly from an existing PCA job
   1. 3-5
2. Enhance the package to have the generic ability to take in grouped data and graph it in an applet (both 2d and 3d)
   1. 3-5
3. Enhance the package to now be able to add user input options to this such as panning, zooming, recoloring, selection, and coloring of classes.
   1. 8-13
4. Create an interface where we can save data from Applet to CaIntegrator2 (such as selected Gene Lists or Sample Lists)
   1. 8-13
5. Total for the features mentioned is 22-31

## Links of Interest

SVN for MEV: <https://mev-tm4.svn.sourceforge.net/svnroot/mev-tm4>

MEV Homepage: <http://www.tm4.org/mev.html>

Gene Pattern Modules SRC code (for PCA Viewer): <ftp://ftp.broadinstitute.org/pub/genepattern/src/GenePattern_modules_src.zip>

## Summary: PCA/TIGR MEV using Java Applets

The best thing about this technology is that it’s in a familiar language, but that’s also a downside because it is older technology (Swing GUI’s in an Applet). Large datasets can be quite slow on the PCA viewer so it would need to be optimized (drawing smaller spheres for datapoints for example). The LOE of getting this implemented with basic features is not very large, because it almost works out of the box with minor code adjustments, but to enhance it would be a bigger undertaking, especially because we’d now be moving to SWING development which hasn’t been used on our current application, but all the features that were requested in the user story can be met using this technology.

# Using Adobe Flex

Adobe Flex is used in the view layer of the MVC architecture to make RIAs (Rich Internet Applications). To develop for this, we would use FlexBuilder 3, which has an eclipse plugin and should integrate right into our project workspace. It can be used in our current tech stack in addition to Java/Struts/Spring by using Flash Remoting with the protocol AMF (Action Message Format, which is a binary representation of the data). So for java we would use BlazeDS to be able to push data in real time to Adobe Flex as well as receive data sent. The other alternative to BlazeDS is LCDS (Live Cycle Data Services), but this is not open source / free, while BlazeDS is. And there’s a framework called “ILOG Elixir” which allows for all kinds of advanced data visualization such as 3d charting and heatmaps to go along with Flex, however, ILOG Elixir costs $799 per license. Another possible free 3d charting library is called “PaperVision3D” with a good example here <http://www.insideria.com/2008/06/3d-charts-using-papervision3d.html> , I was able to use the PaperVision 3D library, but wasn’t able to get this particular source code to work, but that was probably due to my lack of complete knowledge of Flex (just takes more time). PaperVision3D is open source and free, so it would make a good framework for us to create our graphing library on.

## LOEs

1. Learning FLEX well enough to create charts and use Flash Remoting to integrate into our applications business logic
   1. 5-8 (per developer)
2. Integrating Flex into our application stack and making a basic page talk to a remote object, such as getting a list of all subjects for a given study, this requires using BlazeDS.
   1. 5
3. Learn PaperVision3D framework (or ILOG Elixir if licenses could be obtained), and write 3d charting classes with the ability to re-color annotations, lasso tool, pan, zoom, rotate, etc, or find previously created ones such as the one at the link above.
   1. 8-13
4. Link PaperVision3D framework to actual data and have it be interactive with caI2 queries
   1. 3-5
5. Total estimated LOE = **21-31**, but seeing that it’s a new product, the risk is high for this to be on the larger number than on the smaller one.

## Links of Interest

* Flex Downloads Page: <http://www.adobe.com/products/flex/flexdownloads/>
* Adobe Flex Learning Paths Website for Programmers: <http://www.adobe.com/devnet/flex/learn/learningpath.html#type=role&role=programmer&goal=Build%20a%20basic%20data-driven%20application>
* Flex Samples: <http://www.adobe.com/devnet/flex/?view=samples>
* BlazeDS Setup: <http://www.adobe.com/devnet/livecycle/articles/blazeds_gettingstarted_02.html>
* Papervision3D examples with graphs: <http://www.cynergysystems.com/blogs/page/andrewtrice?entry=visualizing_data_in_multiple_dimensions>
* Papervision3D download home: <http://code.google.com/p/papervision3d/>
* ILOG Elixir: <http://www.ilog.com/products/ilogelixir/>

## Flex Summary

It looks like this is a very nice programming language to create rich visualization for CaIntegrator2 and can probably help to create more interactive apps NCI-wide, but there is a learning curve with the technology we would have to ramp up on, and FlexBuilder is not a free SDK. This product does appear to have the ability to do all of the capabilities that we’re looking for however, we’ll have to write most of them.

# Using JavaFX

JavaFX is another language used for RIA (Rich Application Development) and is made by Sun to compete with Flex and Silverlight. One of the advantages that JavaFX has over Flex is that the development environment is free to use. It also uses Java as the client platform, where Flex uses Adobe Flash (most people have both). One disadvantage is that it is relatively new compared to flex (it was released in Dec. 2008, where Flex is about 5 years old and already on version 3). The framework built around **Flex** seems to be much stronger, as is the community support. To transfer data between server and client on JavaFX it would use RESTful web services to send and receive data via XML or JSON formats. The main drawback to how new JavaFX is is that there’s not a lot of 3d work being done on it currently. It does have basic 3d charting capabilities, but nothing very interactive to do the kinds of things we’re doing. One thing I found is where InteractiveMesh is using Java3d inside JavaFX, which would be kind of like a hybrid of using what we currently have (Java3d + Swing in PCA viewer) and JavaFX, but even that product seems in its infancy phase and we would likely be doing it hoping for something better to come along later. A discussion about this can be found here: <http://forums.sun.com/thread.jspa?threadID=5373702> and <http://forums.java.net/jive/thread.jspa?threadID=56033&tstart=0> . I was able to successfully get their “hello cube” to work which is just a basic cube, but wasn’t able to get their more advanced alphabet cube to run. Basically it’s a work in progress and doesn’t have a huge following of people. Based on the lack of community support and the new-ness of the project, it will be difficult to figure out just how hard it is to do 3d graphing, however their 2d graphing is pretty straight forward and should be very easy to use for basic things. One other basic problem I had was in MyEclipse 7.5 the compiler complained about almost every line in the .fx files, even though when I did a “Run As -> Flex Application” it ran just fine, so I’m not sure if there’s an incompatibility issue but I wasn’t able to setup the environment as easily as I was Flex.

## LOEs

1. Learning JavaFX well enough to create charts and use RESTful web services to integrate into our applications business logic
   1. 5-8 (per developer)
2. Integrating JavaFX into our application stack and making a basic page talk to a remote object, such as getting a list of all subjects for a given study, using the RESTful web service.
   1. 5
3. Try to integrate Java3D and FXCanvas3D from interactivemesh or figure out another 3d graphing library and write 3d charting classes with the ability to re-color annotations, lasso tool, pan, zoom, rotate, etc, or find previously created ones such as the one at the link above.
   1. 13-21+
   2. This is a high unknown due to the lack of community support for 3d development on JavaFX
4. Link the new framework to actual data and have it be interactive with caI2 queries
   1. 3-5
5. There’s a high probability that these LOE’s are off due to the inability to find a good working 3d framework in JavaFX
6. The guessed total LOE is 26-39

## Links of Interest

JavaFX Home Page: <http://javafx.com/>

JavaFX Core Scripting Overview: <http://java.sun.com/javafx/1/tutorials/core/>

JavaFX Tutorials: <http://jfx.wikia.com/wiki/Tutorials>

InteractiveMesh 3d add-on: <http://interactivemesh.org/testspace/j3dmeetsjfx.html>

## JavaFX Summary

JavaFX looks like it could be pretty powerful and compete someday in the RIA marketplace, but due to the lack of community support to adopt it might be a gamble without a lot further and in depth research. It might be able to do exactly what we want, but I wasn’t able to quickly find anything as promising as our other options.

# Conclusion

The safest (least risky) way to proceed to generate 3d interactive graphs would be to use the first method of using Swing/J3d with the MEV and PCA viewer code as a starting point. However, there isn’t as much upside to this method because we already know what this will give us and the technology isn’t quite as cutting edge as the other option. The more interesting and possibly better approach would be to use Flex with PaperVision3D on top of it. However, the risk involved here is that it’s all newer technology that no one on our team knows how to use (yet). But that could have been said about Struts2, which we all picked up and integrated well, so it isn’t always best to go with the safest approach. The 3rd option, JavaFX, seems to be too new of a product to evaluate as well as the other two.