Embedding Sirius in a Pure E4 Application

Lessons Learned and Future Plans
About me:

J. Langley -


Developing applications and frameworks based on Eclipse technology since 2009.

Specialize in “Horizontal Integration” - combining existing technology to provide new capabilities.

Find me online:

Email – jlangley@cohesionforce.com
Twitter – @jperiodlangley
About CohesionForce:

Since its founding in 1998, CohesionForce has been providing innovative solutions to our customers’ most complex modeling & simulation, systems engineering, and software engineering challenges.

Core Values:

- Disciplined engineering should be easy
- Systems should be fulfilling to use
- Better tools are needed to solve the problems of tomorrow
Summary:

Ever wonder what it would take to get an Eclipse Sirius diagram editor to work in an Eclipse 4 Application? I took an expedition through Sirius and it's dependencies on a mission to find out!

After a few days of ripping things apart, putting them together, making wild guesses, and more than a few "git reset --hard" commands - I made it to my final destination.
Definitions & Constraints:

**Eclipse 4 Application**: An Eclipse application using the Application.e4xmi and E4 services. There is no Eclipse Workbench, no compatibility layer, etc.

**Sirius Diagram Editor**: This talk involves a “mostly working” diagram editor.
Architecture (As I Understand It):

- Sirius
- GMF
- GEF
- SWT
- Eclipse Workbench
Breaking the PlatformUI dependency:

- Sirius
- GMF
- GEF
- SWT
- Eclipse Workbench

Innovative Solutions For Mission Critical Systems
First Idea:

GEF 4 !!!

After reading some of the news about GEF vs GEF 4, I decided to try switching the GEF dependency used by GMF to GEF 4. I have a small bit of experience with JavaFX, so no worries - right?
First Failure:

I could not find any tutorials or good documentation on migrating from GEF to GEF 4.

It appears that there was an interface used by GMF that was split into two separate interfaces. I did not know enough about how GEF and GMF work to decide which interface to use in which context.

I may try this again in the future after learning more...
First Failure:

```bash
?>git reset --hard
```
Progress Step 1:

1. Cloned the gef-legacy repository.
2. Grep for PlatformUI - You will see this step a LOT!
3. Remove anything that uses PlatformUI

Good news - I only had to modify 12 files to get GEF to run without PlatformUI!

PlatformUI.getWorkbench().getSharedImages().getImage(ISharedImages.IMG_OBJ_FILE);

We should update this article on shared images!
Progress Feels Great!
And then comes GMF:

I almost gave up after searching for PlatformUI in the GMF codebase.

After spending an hour or two, I realized that most of the code using PlatformUI fell into common categories.

We’ll spend a bit of time on each of these:

<table>
<thead>
<tr>
<th>Images</th>
<th>PartService</th>
<th>Logging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>SelectionService</td>
<td>Shell</td>
</tr>
<tr>
<td>Display</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Progress Step 2:

Images, Help, & Display

I commented out anything using the Workbench Shared Images, or changed to use the JFace Image Registry.

Help! PlatformUI.getWorkbench().getHelpSystem().setHelp(parent, helpContextId);

– I commented out all references to Help

Display - changed to Display.getDefault()
Progress Step 3:

Selection, Parts, & Shell

I commented out anything using the Part or Selection Service.

I am amazed that things still “work” after removing some of this code.

The Shell was easier - just followed what I had done with Display.

Logging - `getDefault().getLog().log(...)` - I simply removed these
Almost there...
Getting Sirius:

The Sirius baseline was much cleaner with regard to the PlatformUI references. There were only 11 files that had to be modified to get a running application.

The problems that I encountered were mostly due to behavior to find the active page, context menu, editor, etc. Most of these calls were made through the IEditorSite held by the DDiagramView.
Getting More Sirius:

To provide enough functionality to get this off the ground, I created a class that implements IEditorSite and uses E4 services to provide the needed capability for things like selection listener. This is currently a giant hack, but may prove to be a useful way to get an editor to work without a workbench available.
Now we need an Application:

For the end application, I decided to start with the Arduino Designer from Melanie Bats at Obeo. It’s a cool project with a great goal behind it so you should check it out!

I used the model, edit, editor, and design projects from the Arduino Designer repository. I used the ProjectServices class in the ui project as a guide to create the aird file and semantic model—using file paths instead of relying on the PlatformResource package to create the URIs.
Demo!

I really hope this works!
Thoughts for Future Work:

We now have a Sirius diagram running in an E4 Application!

We also have modified versions of GEF, GMF, & Sirius that will be problematic to maintain as they are modified in the future.

I am looking for ideas for a more reasonable approach to solve this issue - please forward any suggestions that you may have!
Option #1:

One approach would be to split the platform bundle and move the PlatformUI class to a fragment. Then we could create an E4 variant of the fragment that provides an E4 Add-On PlatformUI.

CohesionForce has already migrated some views present in the workbench for use in E4:

https://github.com/CohesionForce/eclipse4-parts
Questions?
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1 0 +1

Innovative Solutions For Mission Critical Systems