Option 3: Compatibility Layer and Eclipse 4 (e4) Plugins

For this option, you would rely on the compatibility layer to reuse all existing components without any adaptations. New components would be developed following the e4 programming model, including dependency injection and annotations. There are three ways to integrate e4 elements into a compatibility layer application.

The first option is to use processors and fragments to add elements to the application model created by the compatibility layer. However, there are currently timing problems. When processors and fragments are being processed, the compatibility layer has not yet created the complete application model. (See this bug report. (<https://bugs.eclipse.org/bugs/show_bug.cgi?id=376486>). Therefore, this **option might work for handles and views, but currently it doesn’t work for editors.**

The second option for integrating Eclipse 4 components is to create a copy of the application model used by the compatibility layer, register it as the application model of your application and add new e4 components to it. The relevant model LegacyIDE.xmi can be found in in the plugin org.eclipse.ui.workbench.

The third option is to use the 3.x e4 bridge from the e4 tools project, developed by Tom Schindl. The goal of the bridge is to ease single sourcing applications on 3.x and e4, which means that views and editors can be used in 3.x and e4 in parallel. To enable this, the plugin org.eclipse.e4.tools.compat provides wrapper classes that implement the interfaces of 3.x. For example, the wrapper DIViewPart implements ViewPart. In the wrapper, you specify a class (POJO), which implements a view following the e4 programming model, including dependency injection. Essentially the wrapper is just a pointer to an e4 object. It will initialize the POJO using dependency injection