This file will attempt to explain the inner workings of Viskit.

To build Viskit, from the command line, type "ant compile." This process will first generate XML binding source files from various Schema files located in the "Schemas: directory. The generated bindings are generated to specific packages in the "src" directory of the Viskit build. This is accomplished via JAXB. XML bindings help to interface java source with the XML representations of EventGraph and Assembly files.

Next, the main Viskit source is compiled. Viskit can now be launched by typing "ant quick.run" on the command line. How to use Viskit is explained in the JavaHelp tutorial that can be accessed from Viskit once it's launched by selecting the Help menu, or by launching the help menu via a command line call to "ant help."

When Viskit is first launched, it creates a .viskit directory in the ${user.home}. Here, configuration files are placed that can be modified from Viskit by selecting File -> Settings. Things such as what panels to expose, the "look and feel" of the UI, size of the UI frame, what project space was last opened, what files were recently opened, and other project history data are recorded. Various logging output files are also generated and placed there for later reference should an error have occurred. All of these can be accessed for content review by running any of the "ant show.*" targets from the command line.

After the .viskit directory is populated, Viskit will popup a dialog to either open an existing Viskit project, create a new one, or exit. A default Viskit project will be created in ${user.home}/MyViskitProjects/DefaultProject if the user attempts to open an existing project, but then cancels. If the user wants to create a new project, the dialog will warn of a project with the same name and give the user another chance to create a project with a different name.

A Viskit project consists of a set of directories and a project config. file:

```
AnalystReports      (main directory for reporting material)
  charts            (JFreeChart charts showing statistical data)
  images            (main folder for screenshots of all Assembly and EventGraph views)
    Assemblies
    EventGraphs
  statistics        (XML files of Simkit generated statistics)
  viskit.ico        (favicon for generated HTML reports)

Assemblies          (main folder for Assembly XML files)

EventGraphs         (main folder for EvevtGraph XML files)

build               (generated java source and byte code class files from EventGraph and
                    Assembly XML files)
  classes
  src

lib                 (directory to hold project specific java based libraries)

viskitProject.xml   (project configuration XML file)
```

The most important feature of the viskitProject.xml file is the MD5 caching of successfully validated and compiled EventGraph files. This is to prevent unnecessary recompilation of project materials once Viskit re-launches and opens an existing project. The MD5 hashes are recorded under the Cached element of the viskitProject.xml file. If a successfully cached and opened EG is modified such that its XML is no longer valid, or doesn't compile, the cached line will be removed from the project's cache element until the issue is resolved. A new MD5 hash will then be generated and recorded in the project's cache element.

MD5 caching is not performed on Assemblies. Assy files are validated and compiled each time the Assembly runner is launched.

If a Viskit project has specific dependencies, jar files can be placed in the "lib" directory of the project and are automatically exposed on the classpath which can be checked by selecting File -> Settings -> Additional classpath entries. Other dependencies can be placed on the classpath when not co-located with an existing Viskit project by selecting those jars via the "+" button which opens a dialog in the parent project directory in order to navigate to and select those jars. The project's "lib" is merely a convenience folder for placing dependency jars. Jar paths are annotated in the extraClassPaths element of the viskitProject.xml file.

A chain of further events is initiated so that Viskit can track EventGraph and Assembly XML file modifications.

1) A DirectoryWatch is first set on a /tmp location for EG and Assy files. A listener listens for the opening of, or changes to these files which ensures that these copies receive any modifications and are checked before saving them back out to the original file. In a nutshell, working changes are tracked here until the file is validated, then all changes are copied back to the orig. files.

2) The nature of the above /tmp file operation is that once Viskit opens, a "lock" file is recorded so that next time Viskit opens, the prior /tmp directory is deleted in order to observe good housekeeping.

3) Next, the project's EventGraphs directory is parsed for the existence of EG XML files. These files are first validated against an EG schema, java source is then generated and placed in build/src, then the source is compiled and the completed byte code is placed in build/classes. Both src and class directories observe package structures. During the first successful save of an EG XML file, its MD5 hash is generated, then recorded in the viskitProject.xml along with the path to its generated byte code (*.class). The build/classes directory is then placed on the live ClassLoader, the same used to compile these EGs.

Simultaneously, when an EG is successfully compiled, its icon symbol is placed on the EG Node palette of the Assembly Editor panel. This is more commonly known as the Listener Event Graph Object (LEGO) tree. Accordingly, the current ClassLoader is parsed for EGs which will also be depicted, i.e. from Simkit, as well as any EGs in "*.kit" jars placed in the project's /lib directory. This allows a user to expand nodes, select EG icons (represented in blue color) and drag to the Assembly canvas to visually construct an Assembly.

Also simultaneously, any PropertyChangeListeners parsed from the current ClassLoader will be depicted on the PCL palette of the LEGO tree under the EG Lego tree as in the same manner as explained above for EGs. The PCL icons are represented with a pink color.

SimEntities and PCLs from other libraries, or class file directories can be placed on the Listener Event Graph Object (LEGO) tree by selecting the "+" button on the LEGO panel. This will allow a user to drag those representative SimEntity icons onto the Assy palette for Assy creation.

```
 * Of importance for the java source generation of an Assembly is the
 * ParameterMap.  The ParameterMap is an annotation construct placed in
 * generated, or hand-crafted Event Graph source code to depict all described
 * parameters of EG constructors which will aid in correct state variable
 * initialization when constructing Assemblies.
 *
 * A zero parameter constructor is always generated in addition to any
 * parameterized constructors.
 *
 * The ParameterMap is also parsed from source code at runtime to allow Viskit
 * to properly identify constructors of EGs classes on the classpath whether
 * they are from generated source, or contained in third party libraries.  The
 * importance of this feature is that iconized EGs will be displayed in the
 * LEGOs tree panel for drag and drop Assembly construction.
```

4) At this time, EG files can be manually opened and modified, or created new. Due to the unique nature of XML IDs, each method within the same EG file must have a unique name to pass XML validation. This is unlike Java which can distinguish between methods with similar name, but differing method signatures. If a new empty Viskit project was just created, a new default EG will be opened. If an EG XML file doesn't validate during a file save, or its corresponding source code compilation reveals an error, the tab that represents the EG will change from a green color (signifying valid XML, good source compilation) to red which signifies an unsaved file, failed XML validation, or failed compilation. The user will be unable to advance to the Assembly Editor tab until the problem is corrected. If an EG's source won't compile, any previously compiled byte code *.class files of the same name will be deleted preventing inclusion on the classpath. A failed compile will also cause removal of the EG MD5 hash entry from the project's viskitProject.xml file. If upon start up, a previously opened XML file passes validation, but fails compilation, the user will again be unable to advance to the Assy Editor tab.

Of note is that to ensure all intended entries are made to each node and edge, be sure to mouse over each and review the tooltip for correct entries.

5) When an Assembly file is opened, the Design of Experiments (DOE) panel is initialized with the Assembly and a /tmp DOE input file is created and passed to the DOE Job Launcher. This file saves the DOE inputs for future use. DOE is an advanced feature and not necessary for pure vanilla Viskit operation.

6) All corresponding EGs are opened in the Event Graph Editor panel when an Assembly file is opened in the Assembly Editor panel. This is accomplished by parsing SimEntitiy elements of Assy files to know which EGs to open. However, if any EGs are subclassed from other EGs, these will not open automatically. Also, if the last remaining Assy file will closed, all open EGs will close including any on-associated EG files that were also opened.

7) At this point, the paths of the any opened Assembly and EG files are recorded in the .viskit/c_app.xml file and will be opened upon next Viskit launch. The last open Viskit project, as well as its history of Assy and EG files will be listed in the File -> Recents menu selections for each editor.

8) Opened Assy files can now be modified, or new Assy files can be created. EG (blue) and PCL (pink) nodes can be dragged across from the LEGOs tree panel, or menu items can be invoked to do the same. EG nodes are parameterized via dialog wizards. RandomVariates are instantiated via the RV Factory static getInstance method utilizing (String, Object...) parameters. Adapters, SimEventListeners and Property Change Listeners (PCLs) are the last to be selected, connected and initialized before an Assy run is initiated.

9) When it is time to run an Assy file, the Initialize assembly runner button is selected from the Assy Editor which fires off a few events:
- a Directory Watch is set for the Assy file
- validate and compile the Assy
- place compiled source on the classpath
- prepare arguments to launch a separate thread for the Assy
- populate the Assy Run panel with replication selections, reset Sim start time to 0.0
- switch panel view to the Assy Run panel

10) When the Simulation Run VCR button is selected, the Assembly is launched via a Thread who's ClassLoader context is separate (new) from the ClassLoader that Viskit is currently running. This is to eliminate any in memory static references to the any previously initialized Assembly parameters thereby giving the current Assembly a clean slate to run an independent sim. The main Thread run method for each XML Assembly file is via subclass of ViskitAssembly which subclasses BasicAssembly which contains the run method. Subsequent selection of the VCR run button will produce the exact same results if nothing in the Assy. has been altered.

11) The simulation can be immediately stopped via the VCR stop button, or it can run until its sim. stop time has been reached for the number of replications selected. Step-through and rewind buttons are implemented, but not working quite right, so, they are disabled upon instantiation from the InternalAssemblyRunner as of now.