**SWMM EPANET User Interface Reengineering Architectural Design**

The reengineering of the two software’s interface will be designed on an open platform that employs open source software and tool chains to help foster the already sizeable user community to continue to grow and develop custom features for these software.

Python is the language of choice for the development work due to its easy and widespread adoption by the world-wide users who are scientists, engineers, software developers in many professions. It is a cross-platform interpretive programming language that has native support on all three major operating systems, i.e. Linux, MacOS, and Windows. In addition, a large pool of open source third party libraries for a great variety of purposes is already in existence and can be readily employed to accomplish many functionalities of the two software ranging from user interface (UI) element to complex numerical analysis.

For this project, the GUI visual layout is created by Qt GUI toolkit (from the Qt Company) via drag-n-drop of UI widgets (such as button, text field, dropdown box, table view etc) onto a design surface (i.e. window). PyQt, consists of Python bindings for the Qt GUI visual elements, handles user interactions with the visual elements via a generic signal/slot mechanism to ensure implementation neutral event handling.

The choice of Python-PyQt framework ensures this project benefit from the combined power of Qt and Python and to render the finished product to be truly cross-platform.

For this project, we choose the PyCharm Community Edition 5.0 from the JetBrains company to be the python source code development integrated environment. This software has a large user community and it is available free of charge.

The overall project architecture is shown in Figure X.

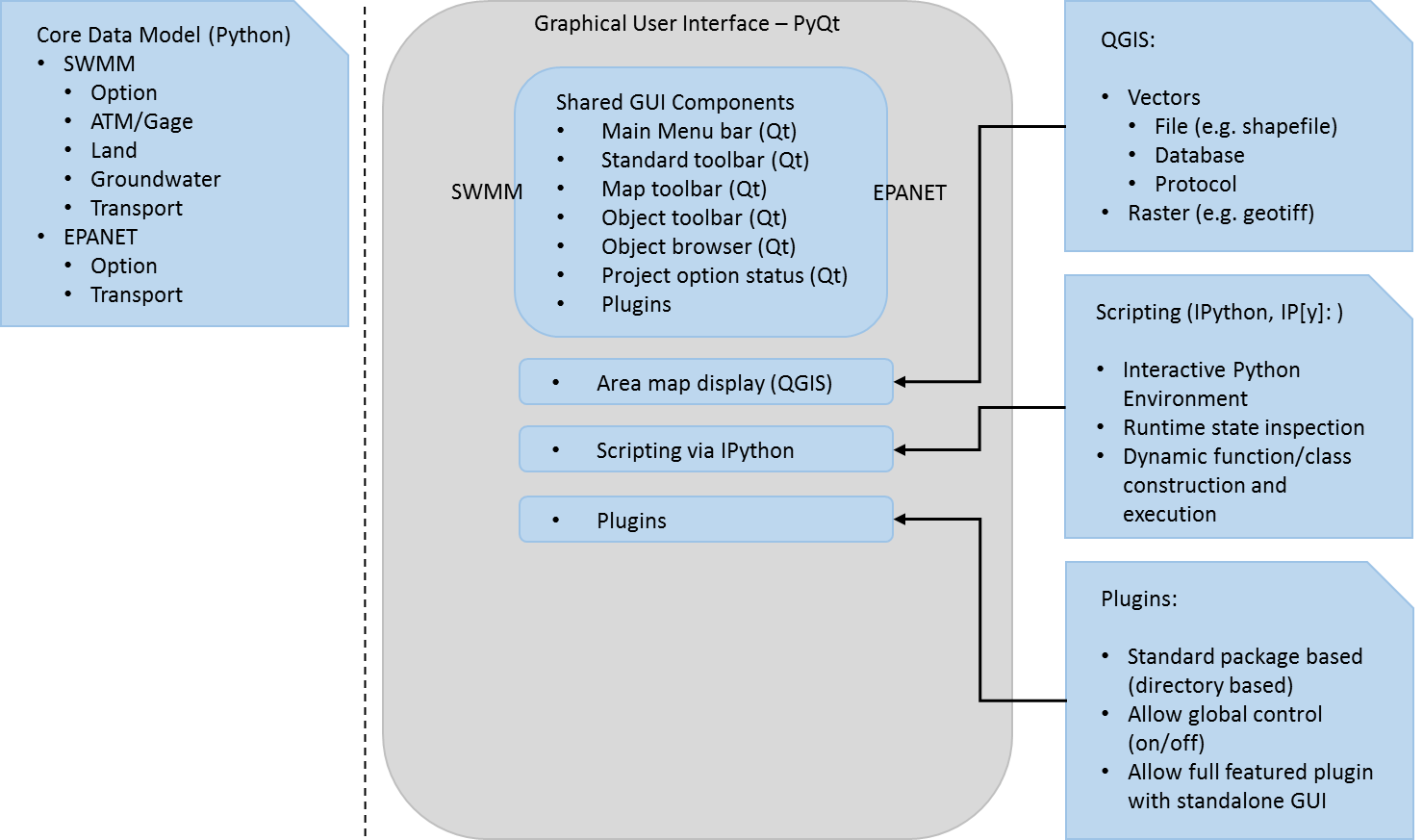
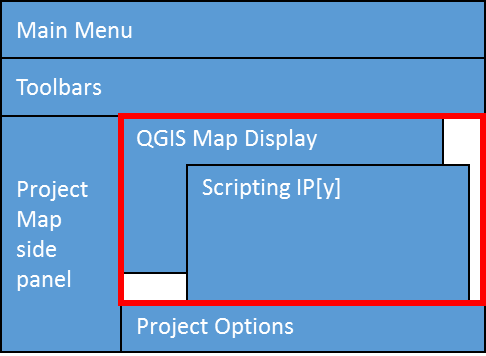


Figure X. Overall project architectural design

The current official EPANET and SWMM software have similar GUI layout and share a common set of GUI control elements as shown in Figure X. Hence, the two software will be designed using a shared architectural framework and the same set of event handling logics in all major UI components, including the three new additional features, i.e. area map display, scripting, and plugins.

The usual UI components including main menu bar, toolbars, project browser, and project option, along with their subsequent chain of UI elements will be designed and function in similar fashion to the current software.

The current area or network map display control will be replaced with open source GIS map display from QGIS (also known as Quantum GIS). Apart from the standalone QGIS desktop application, QGIS is also distributed as core libraries that can be incorporated into other standalone applications such as the reengineered EPANET and SWMM software in this project. The utility of the QGIS library (namely qgis.core and qgis.gui) is accomplished by importing them into the main application at the top of the main form’s python source code. These two libraries provide the main QGIS map control and drawing canvas where the actual GIS data layers will be displayed. A standalone python class (EmbedMap) is constructed as a wrapper of the QGIS libraries to arrive at an independent map control that can be added to any other Qt UI container type of controls, a MDI control in this case, as shown below (red box).



The scripting control is implemented using IPython (a Qt-based python scripting console), which can be activated from the main menu of the application via a ‘Scripting🡪IPython’ menu option. The main program will open an IPython console in a stand-alone window that is within the same MDI control that also contains the QGIS map window (Figure 4). Within the console, users can type in Python commands, function and class definitions, and any other procedures for dynamic execution. Dynamic access of the main program information from within the console can be potentially done via class object sharing on the same executing memory stack.

Plugins are placed in a special ‘plugins’ folder that is made known to the main program during the installation process. They are organized in packages that correspond to the file directory structures, namely, each plugin will reside in its own subdirectory under the ‘plugins’ directory. Each plugin can have unlimited level of packages (or subdirectories) of its own. At the minimum, each plugin must specify three pieces of information, i.e. its name (plugin\_name), whether to generate its own dropdown menu in the main menu bar of the main application (plugin\_create\_menu), and the operations that are to be exposed to (i.e. that can be executed by) end users (\_\_all\_\_ dictionary). Plugins can potentially access main program information via class object sharing at runtime. Upon program launch, the main program will scan the plugins directory for installed plugins and list them in a ‘Plugins’ menu item on the main form of the program, where they can be switched on/off (by mouse click) individually. We envision two major categories of plugins, core plugins and external plugins. Core plugins will be installed with the main program. These plugins offers most desired operations by common model users. The external plugins are those written by individual users that cater to their own specific needs. Peer reviewed plugins can be saved on EPA server for open access by wider audience. The program allows direct placement of plugins inside the designated ‘plugins’ folder to have them exposed to the main program. Potentially, a plugin manager can be constructed to manage the retrieval, install/uninstall of these plugins. This design pattern will provide maximum flexibility to users in constructing their own plugins at various levels of complexity ranging from a single function to an application with its own GUI.

The Data Model class design follows the conceptual model of SWMM and EPANET in terms of representing hydrological compartments and physical processes. A schematic diagram for the Data Model to be used in this project to organize various objects and their properties is shown below:

