Object-oriented application development with MeVisLab and Python

Frank Heckel, Michael Schwier and Heinz-Otto Peitgen

Fraunhofer MEVIS
Universitaetsallee 29
28359 Bremen, Germany
frank.heckel@mevis.fraunhofer.de
michael.schwier@mevis.fraunhofer.de
heinz-otto.peitgen@mevis.fraunhofer.de

MeVisLab is a research and rapid prototyping platform for medical image processing developed by MeVis Medical Solutions AG and Fraunhofer MEVIS.

Algorithms and applications are developed in a visual programming manner by building a network of functional units (modules) where a module represents an algorithm or visualization method for example. Networks can be hierarchically encapsulated into macro modules to form new algorithms or reusable parts of an application. Moreover, MeVisLab offers the possibility to create a graphical user interface (GUI) for an application consisting of various modules using an abstract module definition language (MDL). Dynamic functionality can be added to an application via scripting using JavaScript and Python. This way it is for example possible to react on user interactions, to manipulate the processing pipeline (modules, networks or the user interface) or even to calculate and save results. Furthermore, the application logic is typically implemented using scripting.

In this paper we focus on MeVisLab's scripting capabilities. We will outline the drawbacks of the common functional development approach used in MeVisLab. After that, we present a novel object-oriented approach for scripting of complex applications with Python which overcomes those drawbacks. Both the functional and the object-oriented development approach will be illustrated and reviewed on a simple example application.

Our development concept simplifies scripting of large and complex applications. Communication between modules is done by explicit function calls in the Python script instead of implicit function calls via so called fields and field listeners. This makes communication easier and the program flow becomes clearer. Using our object-oriented development approach, the resulting code is less error-prone and the module is easier to maintain and to extend. Moreover a module’s field interface is reduced making its functionality better understandable for the developer.