Building Geospatial Tools from Workflow Systems

Alexander B. Nagelberg
Louisiana State University
Department of Computer Science
Center for Computation and Technology

September 10, 2007

Triana[1], actively developed at University of Wales in Cardiff, is a powerful workflow manager written in Java. It allows units, small modules performing specific tasks that can take in one or many inputs and provide one or many outputs, to be written in ease and provide workflows with powerful toolsets. Triana provides tools to easily generate templates for creating units in a toolset.

Triana already has some useful toolsets such as audio processing, image processing, mathematical operations, and grid tools. These can be used easily by dragging in objects and connecting them to one another and configured appropriately. For example, if one wanted to save a picture they were looking at on a web site in black and white, they could drag in the *URLImage* unit, connect it to a *ToGreyScale* unit, and finally connect that to a *WriteGIF* unit, run the process, and their picture would be saved from the website to a black and white image in a GIF file (see Figure 1).

The research I conducted at Cardiff University involved creating a new set of tools for handling geospatial data in a workflow, using such resources as Google Maps. The toolset allows geodata to be manipulated between units and even altered in a browser with Google Maps.

The toolset is composed of 6 main tools, and 3 general tools (see Table 1 and Table 2). First, there are two tools that bring in a KML file[2] to a workflow (passed through as a reference to a DOM[3] object) and bring data from a workflow to a KML file (takes a DOM object and outputs to a KML

[1] http://trianacode.org

[2] KML (Keyhole Markup Language) is an XML-based markup language for geospatial data.

[3] DOM (Document Object Model) is a hierarchal structure which can represent XML tags as objects in memory.

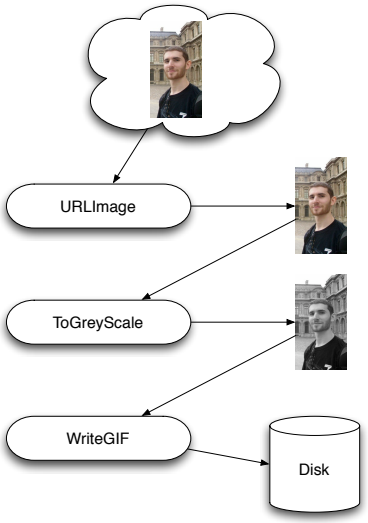

Figure 1: Sample workflow

file – this unit works for XML as well). Secondly, there's a tool which will create a KML object from any query (e.g. my street address). These tools simply bring in a common ground for the workflow – using KML objects.

Table 1: General toolset

TextInput	Introduces a text input to the workflow.
TestOutput	Provides a simple display of XML data from the workflow.
WebBrowser	Introduces a cross-platform web-browser in to the workflow.

There are also tools written that will take a KML object, and import it to Google Maps in the workflow. The user can bring up a browser window in a workflow and simply draw on Google Maps the KML data they imported. Using the other units, this data can then be retrieved from the user's session and either manipulated more or stored in to output KML data.

Finally, there is a tool that will take inputs from either KML data or a latitude/longitude coordinate pair and reverse geocode the data using a REST[4] server from Geonames[5]. That is, it will generate an XML object with information about the place or places given to it. For example, if one

[4]REST (Representational State Transfer) is a simple means of transferring data over a web server.

[5]http://geonames.us

Table 2: Main toolset

ReadKMLFile	Brings a KML file in to the workflow.
WriteKMLFile	Writes to a KML file from workflow.
GoogleGeocode	Passes query from workflow to Google and creates KML object with results for workflow.
GoogleMaps	Passes KML object from workflow on to script hosted on external web server. Script generates a session and passes back to workflow as a full URL to use.
RetrieveKML	Retrieves KML object for workflow via script hosted on external web server. Script pulls data from current session.
ReverseGeocode	Passes either KML object and/or latitude/longitude coordinates from workflow and produces data about the location(s) in an XML object for the workflow.

were to choose a point of interest with the Google Maps unit, it could be passed on to the reverse geocoder and outputted to an XML file which would contain information, if any, about the selected location.

The Google Maps units require a webserver to communicate with. The server-side is done using small php scripts which allow the Triana units to create a session, upload KML data to a session, visualize and manipulate a session, and retrieve the KML data from the session. In the future, these pages could be more dynamic and allow more user input (e.g. letting the user label points or draw and label polygonal regions) and extended to allow custom tiles displayed over or instead of Google's own imagery.

These tools offer definite real-world applications. I am working with the SCOOP(Southeastern Coastal Ocean Observing and Prediction) group at the CCT(Center for Computation and Technology) in developing a group of tools which utilizes the already written ones to offer visualization of weather. This would allow one to view impacts of major storms on major cities using Google Maps.

Acknowledgments

I would like to thank CCT for funding my travel to Cardiff University to do this research. I would also like to thank Omer F. Rana for having me at Cardiff University to work on Triana, Andrew Harrison for mentoring me, and Gabrielle Allen for guiding me on this paper and allowing me to continue this research at CCT.