Toolkit for Automatic Service Generation: KWATT

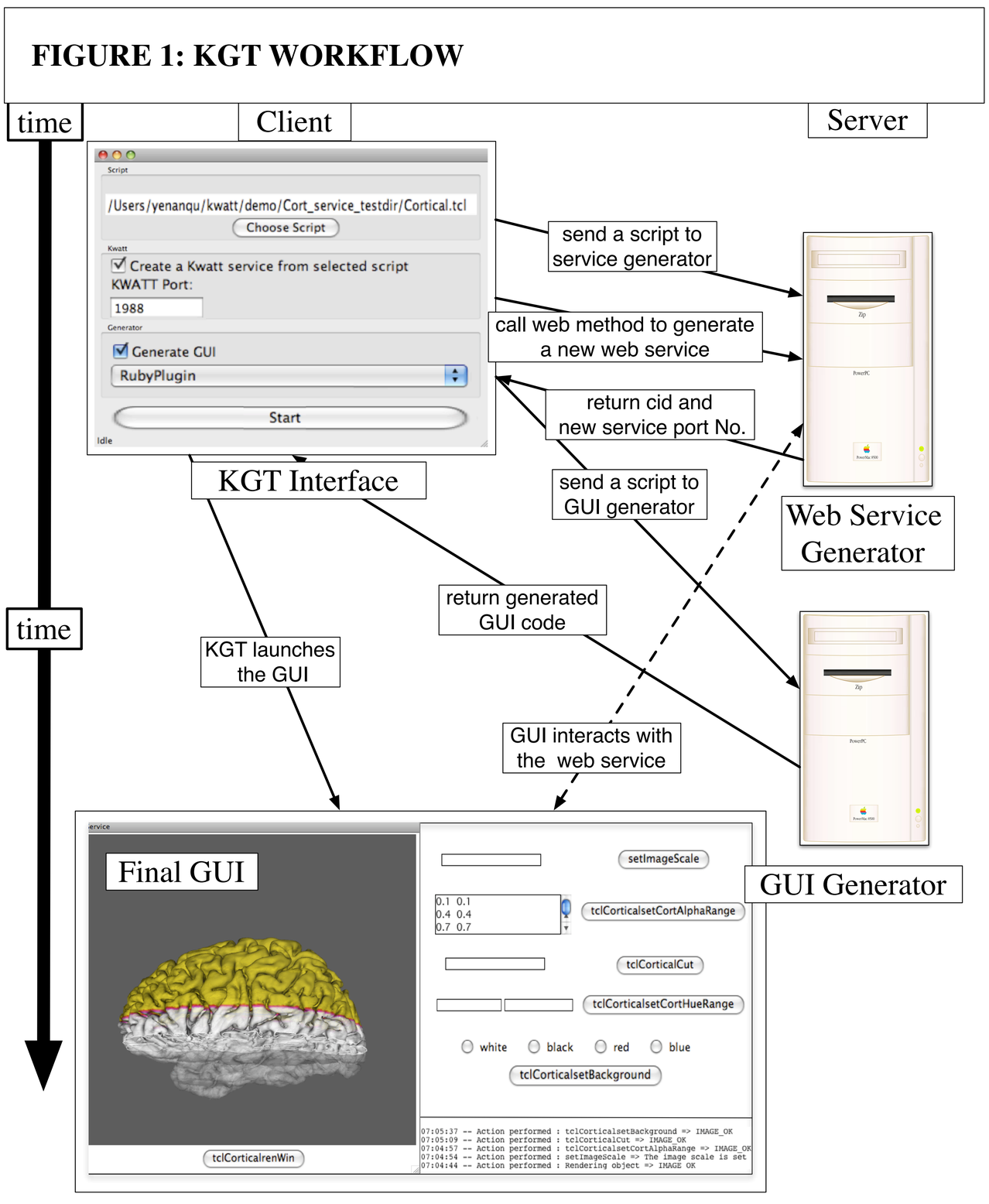
Yenan Qu & Gordon Erlebacher & Evan Bollig
SC, Florida State University
{yenanqu, erlebach, bollig}@scs.fsu.edu

Julien Lafourcade, Magali Lapeyre-Mirande
CS, UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR
julien.lafourcade@gmail.com, magali.lapeyre-mirande@etud.univ-pau.fr

Abstract:
Over the past decade, Web Services have played a prominent role on the World Wide Web and in the business world. Our interest is focused on developing the toolkits for automatic web service and graphical user interface (GUI) generation. In previous work [1], we demonstrated how to translate one or more input scripts into a functional web service, independent of the scripting language. We extend this work by considering the automatic creation of graphical user interfaces to allow interaction between an end user and the web service generated by KSG (*KWATT Service Generator*). KGG (*KWATT GUI Generator*) was developed to achieve this. The KGG is a web service that runs inside the AXIS2 [2] server which is a Java based application, and it performs four major steps of GUI generation. First, the KGG receives the scripts from KGT (KWATT GUI Tools) after the corresponding web service is successfully generated. Comment lines inserted into the scripts provide hints to the XML generator about the interface widgets. Second, the structure of the GUI is encoded into an XML file by parsing those scripts with the XML generator. Third, the KGG extracts information from the generated XML file, then passes them to a plugin. Finally, the plugin generates the corresponding language user interface that is sent back to the user by the KGG.

References:
[1] KWATT: A Toolkit for Automatic Web Service Generation, Yenan Qu, Evan Bollig, Gordon Erlebacher, *Visual Geosciences*, Vol. 13, No. 1. (1 July 2008), pp. 59-69
[2] Official web page: http://ws.apache.org/axis2/

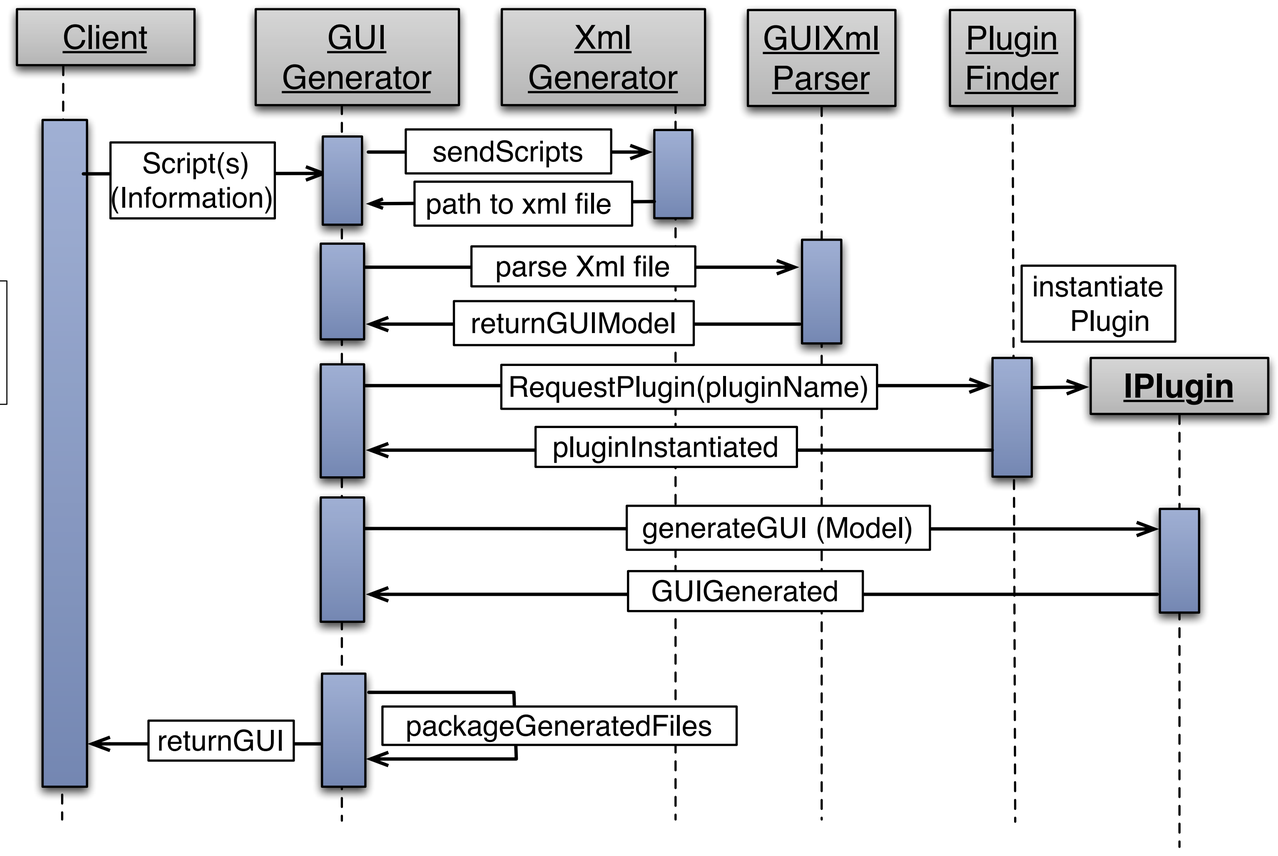
FIGURE 1: KGT WORKFLOW



Both the KWATT Service Generator (KSG) and the KWATT GUI Generator (KGG) run as web services, and are accessed from command line. To simplify management and communication with these two generators, we developed an end-user application: ***KWATT GUI Tools*** (KGT). Figure 1 illustrates the workflow of the KGT. First, users select scripts that need processing, and set up information (such as host name and port name) of the web service and GUI generators via a graphical user interface. When the start button is checked, the KGT sends the selected scripts to the KWATT standalone web service generator, and waits for completion of the generation, then retrieves the corresponding information of the generated web service via a web method call. After the web service is successfully generated, the KGT combines the information from the web service and the source scripts, and forwards this data to the GUI generator. The KGT launches the GUI as it is received from the GUI generator.

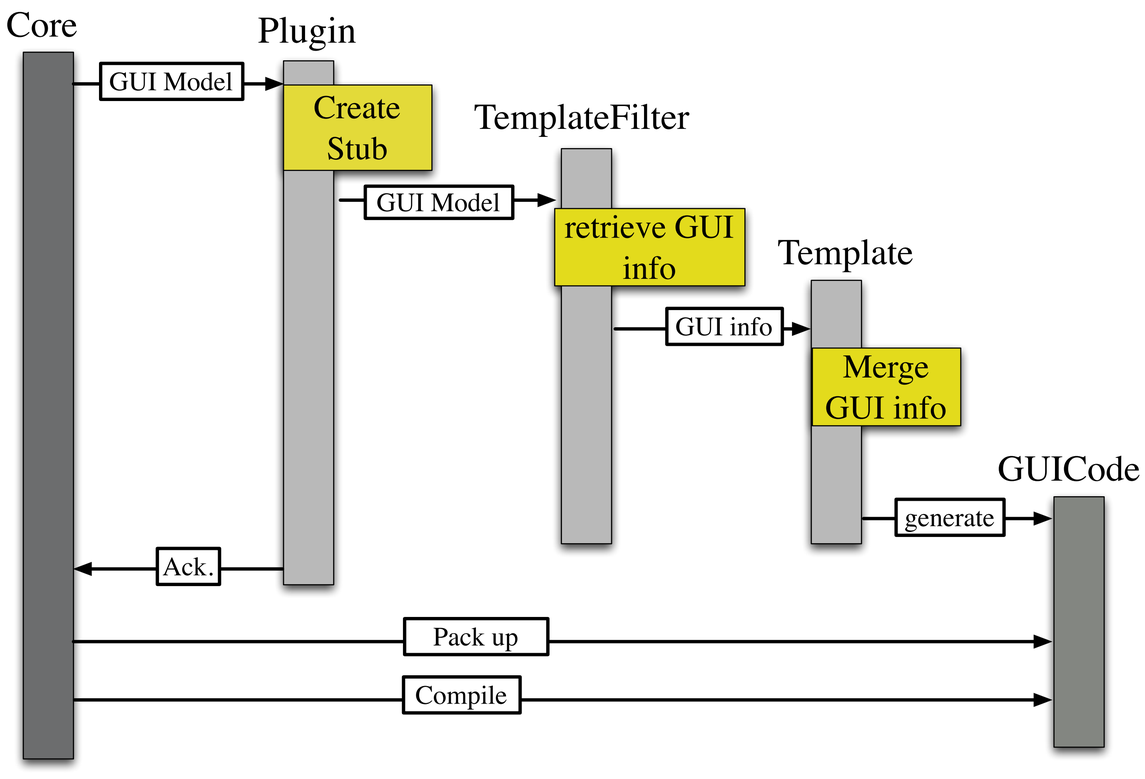
Once the KGG service receives an uploaded script and with the name of one of the available plugins, it runs a system command "xmlGenerator" to generate an Xml file that encodes a detailed description of all the procedures, including parameter names and the various GUI elements associated with procedure parameters. It also includes the URL of the KWATT service WSDL descriptor. The XML file is then parsed by a XML parser and the parser returns a GUI Model. The generator also call a plugin finder to check if the requested plugin exists. If the plugin is found, the finder instantiates the plugin and returns the instance of the plugin. Next, the core of the generator passes the GUI model returned by the XML parser to the plugin, and the plugin sends back a generated GUI. finally, the generator core packs up the GUI source and returns it to the client. As we can easily see, the role of KGG core is like a commander, once receives an input file, it calls corresponding components inside the KGG to process it. Figure 2 presents the flow of data from the time the script leaves the client until the graphics user interface is returned the client.

FIGURE 2: KGG WORKFLOW



Note that the final action for the GUI generation is in a plugin,which creates the GUI code, and the rest of the actions are done in the KGG core. This structure makes the extension and maintenance of the source code more flexible. Since the plugins in the KGG works independently, it is easier to add and remove one of them without affecting other plugins and core applications. Figure 3 illustrates the workflow of a Java plugin.

FIGURE 3: PLUGIN WORKFLOW



Once the plugin receives a GUI model, it passes the model to the template filter, which retrieves GUI information from the model and replaces the corresponding information into the GUI template. The generated GUI code is stored in the directory specified by the core. The plugin returns the instance of the GUI *File* to inform the core that the GUI generation is completed.

Acknowledgement
The authors thank the National Science Foundation for their support through ITR grant NSF-0426867.