Volume Title
ASP Conference Series, Vol. **Volume Number**
Author

Extending Iris: The VAO SED Analysis Tool

Omar Laurino,[1⋆] Ivo Busko,[2] Mark Cresitello-Dittmar,[1] Raffaele D'Abrusco,[1] Stephen Doe,[1] Janet Evans,[1] and Olga Pevunova[3]

[1]*Smithsonian Astrophysical Observatory, 60 Garden St., Cambridge, MA 02138, USA*

[2]*Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218, USA*

[3]*Infrared Processing and Analysis Center, 770 South Wilson Avenue Pasadena, CA 91125, USA*

⋆*olaurino@head.cfa.harvard.edu*

Abstract. Iris is a tool developed by the Virtual Astronomical Observatory (VAO) for building and analyzing Spectral Energy Distributions (SEDs). Iris was designed to be extensible, so that new components and models can be developed by third parties and then included at runtime. Iris can be extended in different ways: new file readers allow users to integrate data in custom formats into Iris SEDs; new models can be fitted to the data, in the form of template libraries for template fitting, data tables and arbitrary Python functions. The interoperability-centered design of Iris and the Virtual Observatory standards and protocols can enable new science functionalities involving SED data.

1. SED Analysis with Iris

Iris provides a set of built-in capabilities that allow researchers to build, view, explore, and model Spectral Energy Distributions (SEDs; Doe et al. 2012). Iris provides the user with a self-contained research desktop where the different components are seamlessly integrated. The tool includes code from mature projects, including Sherpa (Doe et al. 2007) and Specview (Busko 2000), and new code developed for the specific science domain of SEDs. Interoperability with other tools is guaranteed by the implementation of several International Virtual Observatory Alliance standards and protocols.

In this paper we describe the features of Iris 1.2, which is available for download from the VAO website.[1] In the first part we briefly recap the main built-in capabilities of Iris, while in the second part we describe the several extensibility points offered by the Iris Framework and the Software Development Kit.

Iris is currently distributed for Linux and Mac OS X systems, and requires at least Java 1.6. There is also a distribution for Mac OS X 10.5 that is not directly supported,

[1]`http://www.usvao.org/`

and which requires the user to install soylatte, an implementation of Java 1.6 for Mac OS X 10.5.

1.1. Iris Built-in Components

The *SED Builder* component allows users to create SEDs by importing data from several sources and in several different file formats. A dedicated NASA Extragalactic Database (NED) client provides users with the ability to fetch photometric data drawn from the literature about specific astronomical sources. Since NED implements the IVOA SED Data Model protocol, the data are included seamlessly. However, if the user wants to include his or her own data, or if the data come from other VO applications in non-compliant formats, the SED Builder allows the user to import such files by mapping the columns and headers to the IVOA standard Data Model for SEDs.

When a new spectrophotometric segment is added to a SED, the *S*ED Viewer component displays the points in the SED and allows users to explore their metadata via a *Metadata Browser*. This component also provides the user with the ability to filter the SED points according to any information included in the metadata. The viewer also displays the model that best fits the SED, i.e., the results of the *Fitting Tool* that is described below.

The fitting engine provided by default with Iris is Sherpa, which runs as a different process on the user's machine. The communication between the main Iris application and Sherpa makes use of the IVOA standard Simple Application Messaging Protocol (SAMP; Taylor et al. 2012). The Iris desktop provides a Fitting Tool that allows users to build complex models using physical and mathematical components. Different minimization strategies and cost functions are provided, along with the possibility of computing the confidence interval for the best fit parameters.

All the Iris components rely on a low level I/O library implementing the IVOA SED-related Data Models. This library is available as a stand-alone deliverable for Java developers willing to write SED applications using the IVOA standards.

2. Extending Iris: Plugins

Iris is built on top of an extensible framework that was designed to allow third parties to independently develop additional capabilities in the form of plugins, loadable at runtime using the Iris *Plugin Manager*.

The Iris *Framework* is the central component in a layered architecture that effectively builds up a stack of components, where high level components are abstracted from the technical details of the standards and protocols implemented. Moreover, the architecture is loosely coupled, thus allowing components to work seamlessly together, without needing them to be aware of each other's specifications, design or even existence.

Thus, plugins can take advantage of the infrastructure provided by the Framework, in particular:

Menus Components can contribute menu items to the native menu bar, either in the *File* or the *Tools* menu. Furthermore, menu items can be set to have a desktop counterpart, in the form of a desktop button.

SED Manager The SEDs are stored and managed by a SED Manager. Components can interrogate this object about the available SEDs and, if there are several SEDs, which one is currently selected.

Events Given the loosely coupled architecture of the Iris Framework, messages among components are passed as Events. The Framework offers a predefined set of Events to which components can subscribe by implementing a specific callback, but they are easily extendable so that components in the same plugin can take advantage of the same design, if necessary. The predefined events are triggered when SEDs are created, edited, selected and deleted. A similar set of events are triggered when individual segments (spectrophotometric sequences) are created, edited, selected and deleted.

Attachments Components can attach arbitrary objects to the SEDs managed by the SED Manager. This way they don't have to independently manage the additional information they might want to store about the individual SEDs. When SEDs are deleted, the manager takes care of releasing any references to the attachments, so to avoid memory leaks.

3. Implemented Plugins

The following list describes some plugins that have been developed either by the Iris team or by third party developers.

ASDC Data This is the only production-ready plugin developed so far, and was developed in collaboration with the Italian Space Agency Science Data Center (ASDC) of Rome. The plugin provides the user with a rich GUI to access the wealth of curated data hosted by the Data Center, including time information about the photometry points. The GUI allows to make a positional query in terms of space (RA and DEC) and time coordinates. The ASDC plugin is currently distributed with Iris inside the 'contrib' directory.

R Integration We have experimented with advanced inter-language integration with R, the statistics language and environment. The plugin is just a proof of concept that shows how users might write callbacks in R that can be executed when Iris Events are triggered, e.g., when a SED is created or a segment is added to it. The SED is replicated on the user's R workspace for interactive analysis.

Vizier Client An experimental component was created to query the photometric data provided by the CDS Strasbourg service Vizier.

4. More Extensibility Points

Iris offers a fair range of extensibility points, briefly described in this section.

Custom File Filters Iris supports a fair number of file formats natively: VOTable, FITS, CSV, TSV, ASCII, and IPAC tables. However, new file filters can be created and loaded at runtime. One can also create filters for the natively supported files. In this case, the custom filter would parse the file and map the metadata to the IVOA Data Model fields.

Persistency Components can also get a handle to the configuration directory (usually a hidden folder in the user's home directory) if they need to persist information like user's preferences or work sessions.

Interoperability The Iris desktop takes care of the SAMP connection and the SED Builder implements some SAMP handlers itself in order to exchange messages with other VO-enabled tools. However, components in external plugins might want to add their own SAMP handlers, or send SAMP messages themselves. The Framework provides some shortcuts to do so, which make interoperability very easy to implement, with only limited knowledge of the technical details of the SAMP protocol.

Command Line Features Some components can implement some command line features to Iris. The Framework will parse the command line passed to Iris and relay the arguments to the component. This feature assumes a stateless invocation from the command line, but it is feasible to provide more complex capabilities by taking script files as arguments. The SED Builder itself provides such a capability in order to allow users to import files in bulk and build SEDs from shell or other languages' scripts.

5. How to Develop a Plugin: The Software Development Kit

In principle, developers can use the Iris-common JAR file, hosted by the VAO artifact repository, to freely develop their plugins. In the end, an Iris Plugin is an implementation of some Java interfaces and abstract classes.

However, a more effective and efficient way is to use a specific Maven archetype to create a Java project, complete with build scripts, dependency management and a full example of a test plugin. This test code can be easily edited to set the proper metadata and implement the framework callbacks. At the very least, custom components need to provide one menu item and implement its callback, which is invoked when the user selects the menu item or clicks on the corresponding desktop button.

More complex plugins can implement several integrated components contributing command line features, additional SAMP handlers, and so forth.

Acknowledgments. Support for the development of Iris is provided by the Virtual Astronomical Observatory Cooperative Agreement AST0834235 with the National Science Foundation. Individual components have also been supported by the National Aeronautics and Space Administration (NASA) through the Chandra X-ray Center, which is operated by the Smithsonian Astrophysical Observatory for and on behalf of NASA under contract NAS8-03060, and by the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Inc., under NASA contract NAS5-26555. This research has made use of the NASA/IPAC Extragalactic Database which is operated by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the NASA.

References

Busko, I. 2000, in Astronomical Data Analysis Software and Systems IX, edited by N. Manset, C. Veillet, & D. Crabtree (San Francisco, CA: ASP), vol. 216 of ASP Conf. Ser.), 79

Doe, S., et al. 2007, in Astronomical Data Analysis Software and Systems XVI, edited by R. A. Shaw, F. Hill, & D. J. Bell (San Francisco, CA: ASP), vol. 376 of ASP Conf. Ser.), 543

— 2012, in Astronomical Data Analysis Software and Systems XXI, edited by P. Ballester, D. Egret, & N. P. F. Lorente (San Francisco, CA: ASP), vol. 461 of ASP Conf. Ser.), 893

Taylor, M., Boch, T., Fitzpatrick, M., Allan, A., Paioro, L., Taylor, J., & Fay, J. 2012, Simple Application Messaging Protocol, Version 1.3, Tech. rep., IVOA Recommendation