A new GUI system for ASPRO

Mella G.[a] and Duvert G.[a]

[a]Laboratoire d'Astrophysique de Grenoble, BP 53, F-38041 Grenoble Cedex 9, France

ABSTRACT

ASPRO (Astronomical Software to PrepaRe Observations) is a software tool built and maintained by the Jean-Marie Mariotti Center (JMMC) that provides means to prepare and test the validity of observations on various existing interferometers, noticeably the VLTI. Concerning our web development of ASPRO, our new generic GUI system is a fast light solution to bring a GUI to the applications or languages that haven't such capability. The toolkit is conceptually divided into three parts. The main application is considered as a server. The client handles widgets and graphic presentations and the user interactions according to the applications. One gateway system manages the data flow. Then messages are generated by applications and addressed to the client. The client presents informations to the user and returns back values or commands. This paper describes the overall architecture. It details the application's interfaces. It lists widgets capabilities and graphic functions. It presents features of our first JAVA client which supports XML exchanges. Finally we present how several distant applications are linked to one client but also between themselves.

Keywords: graphical user interface, connected client/servers mode, XML, SVG, network socket, lightweight distant display, consistent variables

1. INTRODUCTION

ASPRO is a tool for preparing observations with current and future optical interferometers. A new GUI system has been developed to allow any astronomer to use ASPRO through a web interface. The main goal of this toolkit is to provide a fast and light remote GUI. The approach used is based on XML document exchanges and an engine to build dynamic windows. No application specific process is done into the GUI engine, the GUI only displays plots and widgets, and returns user's events. The network bandwidth is dedicated to widget descriptions and user events.

2. QUICK OVERVIEW OF RELATED TECHNOLOGIES

Actually ASPRO is able to be run under three modes:

1. Stand-alone.
2. Java applet.
3. Java WebStart [*].

This sections briefly presents the related technologies of the different components.

Further author information: (Send correspondence to Guillaume Mella)
Guillaume Mella: E-mail: guillaume.mella@obs.ujf-grenoble.fr

[*]http://java.sun.com/products/javawebstart/

2.1. ASPRO Related Technology

ASPRO is a stand-alone application. It runs on most linux-like systems. The main features of ASPRO are:

- one command interpreter.
- prompt input or GUI.
- a fast graphical engine with modular rendering.
- a rich set of scientific procedures.

Concerning the GUI, a previous version was compiled with Motif or XForms libraries. Today it is easy to switch between previous libraries and new one using a simple environment variable. The new one is developped to separate the GUI part from the main software core. In fact, this new library does not cover all features to build a real GUI. However it gives a good abstraction layer based on XML. To be complete, the GUI system needs a client which transforms the XML descriptions into graphical elements and user events into XML documents.

2.2. New GUI System Related Technology

The new GUI system is mostly based on modern technologies which are XML[†] and SVG[‡]. Communication is done over TCP/IP sockets between ASPRO and the GUI.

2.2.1. Improved ASPRO

When ASPRO is toggled to use the new GUI library, ASPRO only handles XML exchanges with the GUI system. One new SVG renderer has been written to produce a new output format. A Portable Network Graphic (PNG) library is used to integrate non vector graphics.

2.2.2. The Real GUI

Our first running GUI was developped in Java 1.1 . To handle XML messages, one parser has been written entirely to be light and fast. SVG handling was also developped from scratch according to the supported set of the ASPRO's graphical library. To take advantage of Java 2 image API improvements, a new version has been developped using Java2 virtual machines.

3. SYSTEM ARCHITECTURE FOR MONO-USER CONFIGURATION

In this configuration a minimum of three components must run:

- The main software.
- The GUI sytem or GUI client.
- A messages router or gateway.

The first activated component must be the gateway because it is the communication server. Then the main software and the GUI client connect the gateway using TCP/IP client sockets.

3.1. Communication Management

The gateway is the central point of the XML messages exchanges. Each connected entity must begin to register itself. Registration consists in a XML message transmission which gives information such as identity name. Then each socket managed by the gateway is linked to a corresponding application.

[†]http://www.w3.org/XML/
[‡]http://www.w3.org/Graphics/SVG/

3.1.1. Message Routeing

After identification level, each identity can send a message to a remote application using `from`, `to` and `replyTo` attributes for the root node of the document. Then the gateway forwards the document to the right socket. In simple cases, most exchanges are made between the GUI client and the main application. However a second application can be launched using an already started GUI client. Then the GUI client manages all related to both applications. Sect. 3.4 details mechanisms used to run one GUI client with several applications. Because all tasks are entities, the gateway provides to applications communication facilities between themselves.

3.1.2. Link Loss

Because each application handle communication using client sockets, there is no way to detect at the socket level, that the link is lost on other entities. However by transmitting configuration documents to the gateway, commands can be triggered after the link loss event. In the same way, XML documents can request the gateway to list all registered entities.

3.2. File Management

Considering the mono-user configuration, the filesystem of the GUI client is the same as the application's one. Again, a filename given by the client will be accessible automatically by the application. It is convenient from the user point of view to place a new file in its account and to make the application use it. The problem is much more different in the server configuration, see Sect. 4.2.

3.3. XML Documents Management on the Application Side

In all cases the gateway must get XML documents from the entities. The consequence is that the application should produce XML documents to generate a GUI description which will be engined by the GUI client. At the present time, the application also should be able to get XML documents describing user events. It would be efficient to be able to transform dynamically such XML documents into application level commands. XSLT would do this job according to defined rules of transformation.

3.4. XML Documents Management on the GUI Side

3.4.1. Internal message routeing

We decided to implement a subscribe mechanism to the internal architecture of the GUI client. Then one subcomponent which must handle certain documents subscribes during GUI client inialization, and receives automatically the document to process when document's rootnode tag matches. Such mechanism makes the internal routeing clear and very easy. Several parts can subscribe the same document type and will receive notification at the same momemt.

3.4.2. More than one application management

Sect. 3.1 indicates that the GUI client should handle interface for several applications at the same time. The GUI client always knows which application produces the document it receives. Then each user event related to, e.g., windows, will be returned back according the `replyTo` attribute of the XML source document. Responses to application are done into three steps:

1. document is prepared appending the right `to` attribute.
2. document is transmitted to the gateway.
3. gateway forwards document to the right application.

3.5. Variable management

Because the application and its GUI can be on two different systems, the variable consistency must be handled by the GUI client. Variables are associated to widgets and must be named. Two variables with the same name are considered different by the GUI client if they does not share the same namespace which is the application name. Considering one application, two widgets with the same variable name present into two different windows are considered as the same variable. When the application changes the value of the named variable, the GUI client should be informed. Then the client updates all widgets named by the given id. We decided to inform the application that the user changes a variable only before an action event. In this way the exchanged messages are reduced (take for example slider widgets). It could be a lack of feature to update values on this way. Such need must be identified and could be integrated into the next XML widget descriptions.

4. SYSTEM ARCHITECTURE FOR SERVER CONFIGURATION

Using this configuration, remote client can access our server. The GUI client only require cpu and memory to handle windows and graphics. All processes are done into the application and by consequence on our server. One of the major side of this configuration is, that powerful softwares can be used through a simple web browser. The actual drawbacks are:

- server should assume computing for many users and has a maximum number of sessions.
- processes are done remotely, so user provided file, must be uploaded to the server.
- users must be connected to the network during all the session.

However real benefits are:

- No installation is required excepted for the Java2 environment.
- The server upgrades are shared with every users without user action.
- Required bandwidth is very thin due to GUI description handled by the engine of the GUI client.

4.1. Client connection management

Xinetd[§] is used on the JMMC's server to handle Java applets or WebStart connections. Xinetd configuration is very powerfull and fully delegates connection handling. Client access policy is very wide and allows a simple server configuration at high level. But the main job it does is to start a script launching a new instance of the application after a client connection. We decided to make servers run by different users so their data can be managed separately.

4.2. File Management

At the present time nothing has been implemented to link the user's filesystem and the server one. However, the user can upload his files into a shared area on the server using the POST method of the HTTP protocol. He also can use URL as filename which makes the application get the remote files.

5. GUI CLIENT FEATURES

All features presented in the following section are provided using different XML documents produced by ASPRO (or any kind of application dealing with the GUI subset of XML tags).

Our Java client handles each presented feature. If one other client implementation would like to interface with the system, then this new client is in charge of displaying all messages generated by the core-applications. Each section indicates how to use the associated document and sometimes how our Java client handles some features.

[§]xinetd is an extended Internet services daemon

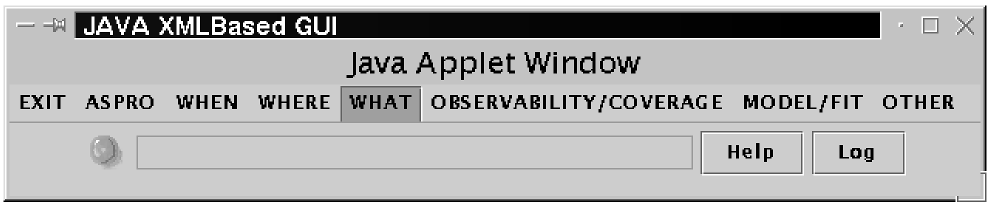


Figure 1. Example of a menu bar displayed by the Java GUI applet

5.1. Menu bar

This element is one central point of the GUI because it generally lists all features of the software. Concerning our Java client, if the client can touch more than one application, the menu bar switch from the horizontal presentation to the vertical one. Then the main menu will use application name and each entry will display it's own menu. See Fig. 1 for an example of a simple menu.

5.2. Status display

This information informs the user about processed action. Success or failure action can be displayed. Reason can be appended to the application's message to detail informations.

Our Java client uses this document to trigger the status icon and to fill the status field. See Fig. 1 for an example of a status icon presented near the menu.

5.3. Logging and command prompt

Application can output information on a logging windows. Application can also get commands directly typed from a command prompt. The application can change the prompt text and color.

Our Java client places logging informations and prompt in a dedicated windows. If more than one application is accessible , tabbed panels contain informations relevant to each application.

5.4. Windows

Window creation is in charge of the application and corresponds to a new description of window sent by the application. After creation the user can close the window. Because all windows are named, any application can be informed when the user closes a certain window. Moreover the application can remotely close a window depending on conditions like user events for example. A multiwindows mode can be deactivated to avoid too many windows on the user's side. See Fig. 2 for an example of window containing various widgets.

5.5. Widgets set

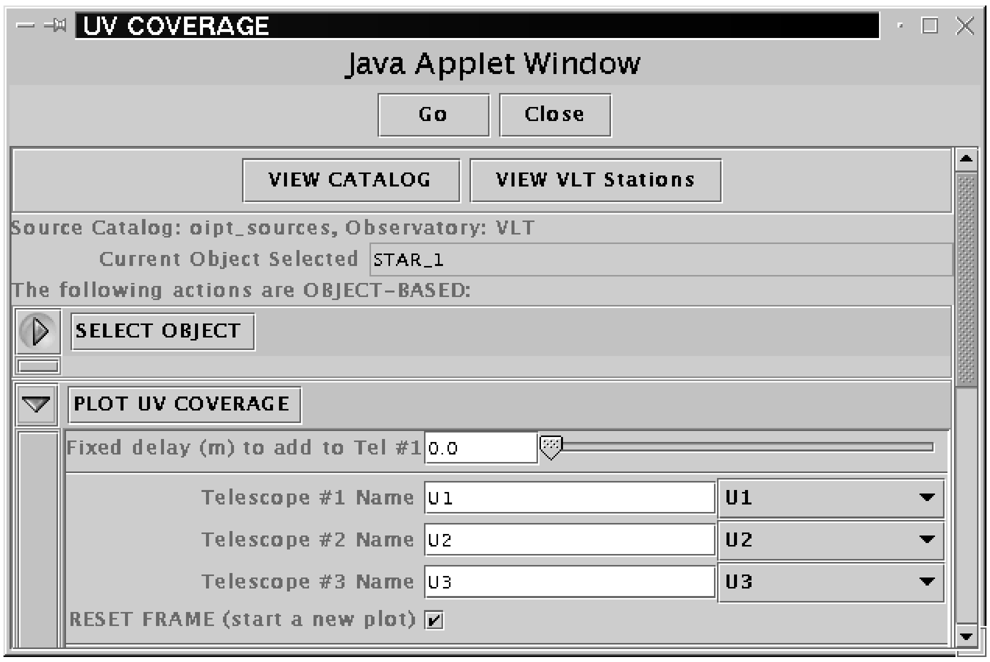
A basic widget set has been choosen for first developements. Each widget is able to display related help message. This section presents all of supported widgets. See Fig. 2 for an example of a window containing some widgets.

5.5.1. Panel

A panel is used to hide a group of widgets behind a simple line. The subpanel is deployed by clicking an extand button. A command can be associated to the related content. Sect. 3.5 describes how underneath variables are handled.

5.5.2. Button

A button simply returns a command to the associated application of the window. At the present time no application code is executed into the client after button clicks. The consequences of the command always are suggested by the remote application side.



Source Catalog: oipt_sources, Observatory: VLT
Current Object Selected STAR_1
The following actions are OBJECT-BASED:
Fixed delay (m) to add to Tel #1 0.0
Telescope #1 Name U1 U1
Telescope #2 Name U2 U2
Telescope #3 Name U3 U3
RESET FRAME (start a new plot) ✓

Figure 2. Example of widgets included in a window displayed by the Java GUI client

5.5.3. Chain

A chain is a string input field. It can be editable or not depending on one XML attribute value.

5.5.4. Label

A label is a readonly string. It only helps the GUI comprehension.

5.5.5. Separator

Like labels, a separator helps the GUI presentation and understanding.

5.5.6. Check Box

A check box is used to handle a boolean variable.

5.5.7. Choice

A choice widget is used to point over a specific value. This widget works in one of the three possible modes. Two attributes are associated to the widget. It can be editable or not and must return content value or index.

5.5.8. File browser

At the present time, only a mono-user configuration can use this widget to get a filename browsing the local filesystem. Sect. 3.2 describes in detail the file management.

5.5.9. Slider

A slider represents a numerical value. The minimum and maximum values given by XML attributes must be respected by the GUI. Like most other widgets, a default value can be given in XML description.

5.5.10. Table

A table helps to present arrays of data. At the present time, data are read only.

5.6. SVG Graphics

ASPRO actually uses graphics, even though SVG is very verbose. By using a subset of elements, SVG is very easy to handle using all kind of graphical libraries. SVG main interest is that it transfers all the plot information to the user GUI without loss (zoom) . Moreover it can produce high resolution plots with a small amount of memory requirement. Our Java client handles zoom and anti-aliasing function. Moreover each window can mix widgets and SVG plots inside a single window. Fig. 3 points out a graphical plot example.

Here is the supported SVG subset of our Java client:

- g (for grouping)
- text
- image
- line
- circle
- rectangle
- polygon
- polyline

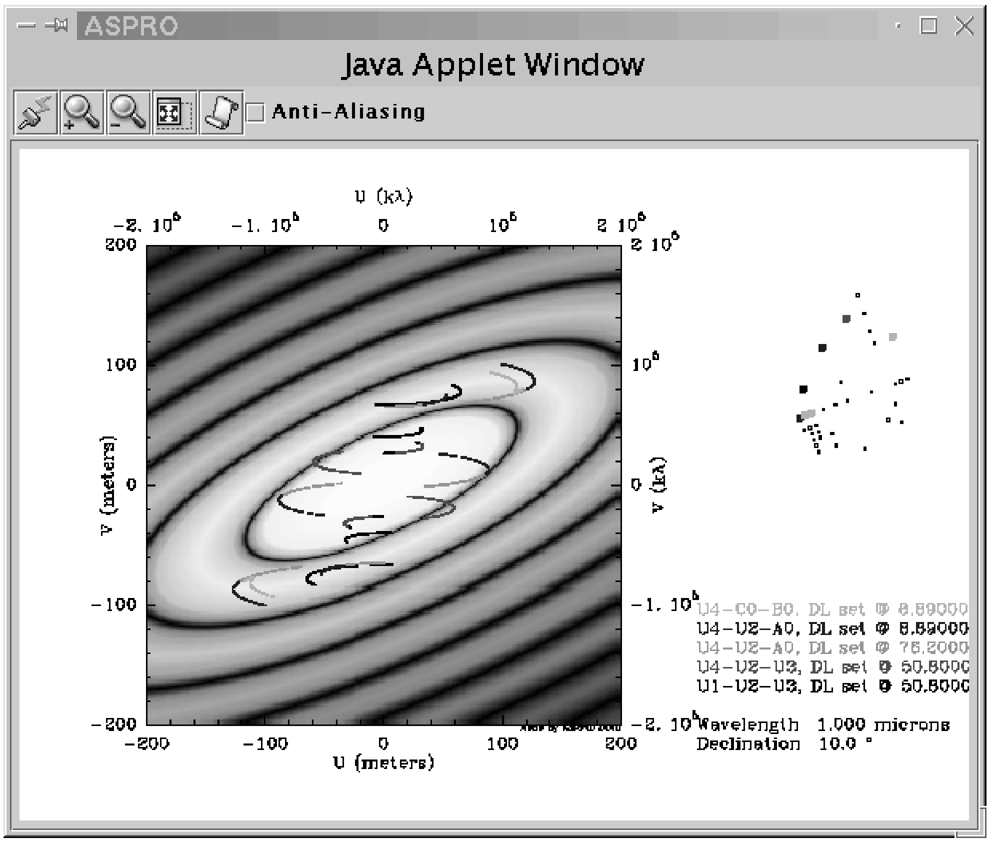
Additionnaly, all of these elements can get stroke or color attributes.

5.7. URL Display

Application can indicate the client to load an URL using an external web browser application. Our Java client uses the browser which loaded the applet or the defined browser of the Java WebStart environment to display the linked document.

6. DEVELOPMENT STATUS

The new GUI system has reached a satisfying level of use. 16 sessions in parallel are available on the server since 12 past months. Concepts are approved. Due to the evolution of the computer power, some technical solutions are requested to change. Because some parts required Java2, there is no reason to maintain a light but poor XML parser. We will use the XML parser included into the core classes of Java2. In the same way, several SVG projects in Java are really good stuffs. We will integrate one of them in our future release of the GUI client. To make interfaces much simpler, XSLT capabilities will be included into the gateway system. To make clearer both server and client filesystems, from the user point of view, we will develop a two containers filemanager. To enhance the GUI use, some user configuration parameters will be managed. Widgets placement will be incorporated into the new XML document definitions.



U4-C0-B0, DL set @ 8.89000
U4-U2-A0, DL set @ 8.89000
U4-U2-A0, DL set @ 76.2000
U4-U2-U3, DL set @ 50.8000
U1-U2-U3, DL set @ 50.8000

Wavelength 1.000 microns
Declination 10.0 °

Figure 3. Example of a plot displayed by the Java GUI client