Java for Astronomy: Software Development at ESO/ST-ECF

M. DOLENSKY[1], M. ALBRECHT[2], R. ALBRECHT[1], P. BALLESTER[2], C. BOAROTTO[2], A. BRIGHTON[2], T. CANAVAN[2], M. CHAVAN[2], G. CHIOZZI[3], A. DISARÒ[2], B. KEMP[2]

[1]ESA/ST-ECF; [2]ESO/DMD; [3]ESO/VLT

Overview

The first part of this article discusses the Java computing language and a number of concepts based on it. After learning why Java is best suited for many applications in astronomy including ESO's JSky initiative and certain Web services there is a second part containing a collection of application briefs from software projects at ESO and ST-ECF. Appended is a list of contact points in house (Table 1) and an applet gallery (Table 2).

1. Why Java?

Here are the key features [1] that the authors felt were the most important arguments why Java was the first choice for their projects:

• Portability: There is only one version of source code and one version of binaries for all operating system-architectures and hardware platforms. As a result of the architecture-neutral capabilities, Java programmes execute on any platform for which a Java Virtual Machine (JVM) exists (e.g. Windows, Solaris, Mac OS, and Linux). Some web browsers come with a built-in JVM.

• JavaBeans Component Model = built-in support for code re-use: Java incorporates JavaBeans which is a software component model. The component model describes how to build an application from software components. In contrast to software libraries Beans provide an additional interface in order to support their re-use and customisation. The JavaBeans concept is widely supported by development tools.

• Common look and feel: There is built-in support of powerful GUI widgets (Swing classes) and common look and feel on different platforms.

• Applets: Applets are programmes running via web browsers. Therefore, no installation is necessary on the side of the web client and also no maintenance. Computations are performed on the client computer thereby off-loading our servers.

The next two bullet lists summarise more concepts and advantages as well as the weak points of Java technology.

More advantages and supported concepts

• Remote Method Invocation (RMI): RMI allows the execution of tasks on other machines.

• Jini: Jini defines a protocol for communication with devices. A device is, for instance, a disk drive, a TV, VCR or cellphone. With Jini there is no need for separate driver software. It's a very general implementation of plug and play capabilities.

• Enterprise JavaBeans (EJB): EJB is a specification for server side components based on JavaBeans. EJB components neither deal with server-side system-level programming nor with client-side interfaces and therefore form a middle tier containing business logic only. This gives much more flexibility when coping with changes of business rules.

• Lots of free software and a lively community with newsgroups

• Object-oriented: This implies encapsulation of data in objects and support of code-reuse.

• Security: Various security mechanisms prevent malicious code from being executed.

• Multi-threaded: Java supports thread synchronisation and enables an application to perform several tasks in parallel even if the underlying operating system is not capable of multitasking.

• Late binding: Java is capable of dynamically linking in new class libraries or instance variables at runtime.

• Robustness: Strict type checking, no pointers, no explicit memory (de)allocation, enforcing boundary checking when accessing arrays

Applet specific advantages

• No installation: Just requires a Java enabled web browser.

• Automatic software update: The latest software version is loaded whenever an applet is invoked. There exists only one copy of the software on the web server.

Weaknesses

• Rapid development led to a steady stream of new Java versions and framework specifications rendering it a moving target. It is only slowly settling down.

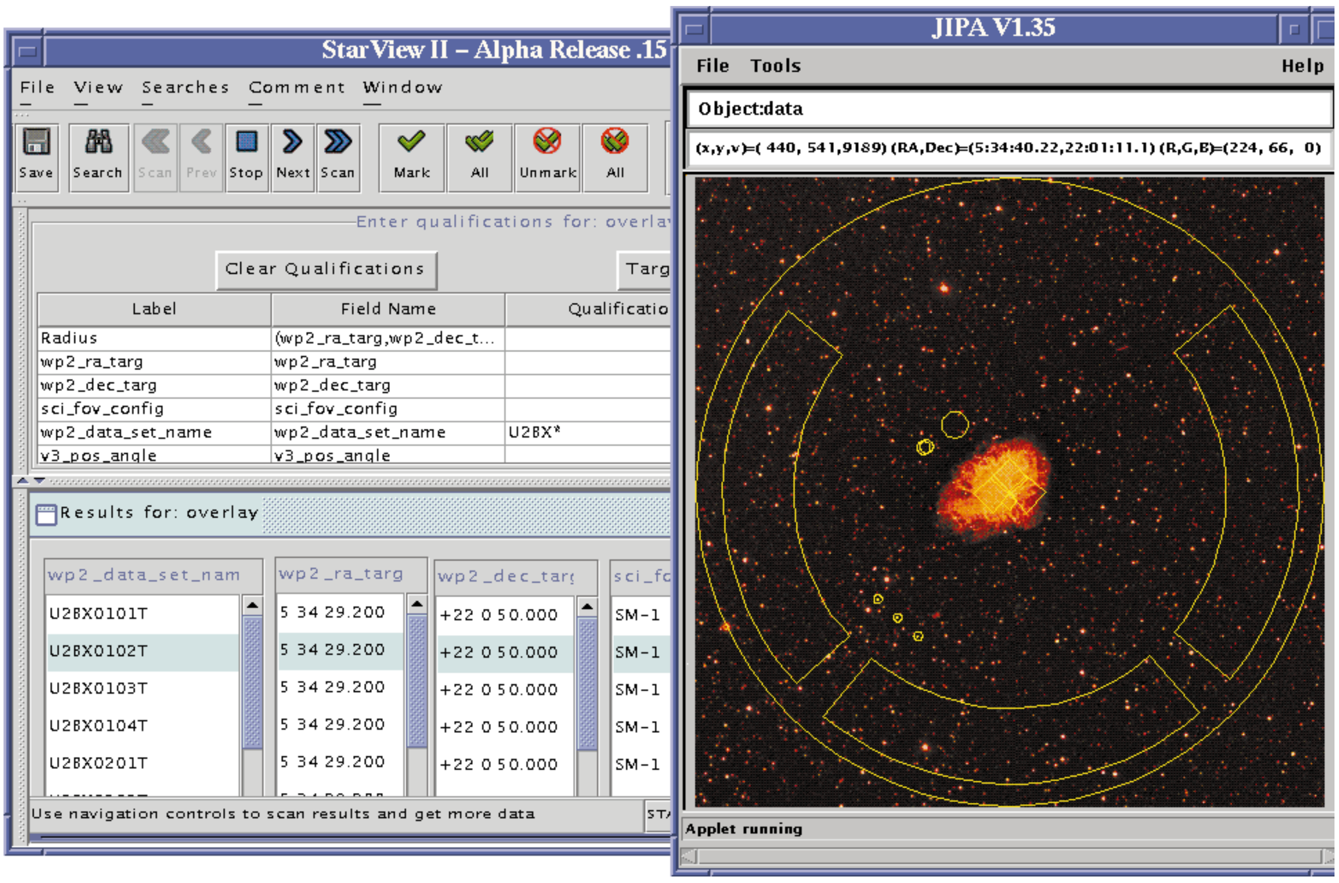
• Write once but install (in case of applications) and test everywhere: Platform specific bugs such as incompatibilities between JVMs are hard to detect and circumvent.

Table 1: Contact Points and Java Links in House.

Service	Contact Point
Java Interest Group (JIG) Mailing List	jig-list@web3.hq.eso.org Send mail to Tim Canavan tcanavan@eso.org for registration.
Java tools in archive and dmd UNIX domains	Bob Kemp bkemp@eso.org
Java tools for archive operations and ecf domain	Markus Dolensky mdolensk@eso.org
JSky Home Page	http://archive.eso.org/JSky/
JSky Mailing List	jsky@egroups.com Send mail to jsky-subscribe@egroups.com for registration.
JSky Repository	ftp://ftp.archive.eso.org/pub/jsky
P2PP Project Page	http://www.eso.org/~amchavan/projects/j p2pp/development-docs.html

Table 2: Applet Gallery.

Applet Name	Description	URL
ESO ETC	ESO Exposure Time Calculator	http://www.eso.org/observing/etc/
Isql	JDBC interface to databases	http://archive.eso.org:8009/ sample/gateway.html
JIPA	Download Page of Java Image Preview Application	http://archive.eso.org/java/jipa1.35/
JPlot	Applet for Jitter Analysis	http://archive.eso.org/archive/ jplot.html
Spectral	HST Preview Applet for Spectra	http://archive.eso.org/preview/ preview/preview_hst/Y29E0305T/ fits/spectral/4
TimeGraph [10]	Applet for System Load Monitoring	http://www.hq.eso.org/bin/ tg?schema=sysload


StarView II – Alpha Release .15
File View Searches Comment Window
Save Search Scan Prev Stop Next Scan Mark All Unmark All
Enter qualifications for: overla
Clear Qualifications
Targ
Results for: overlay
JIPA V1.35
File Tools Help

Label	Field Name	Qualificatio
Radius	(wp2_ra_targ,wp2_dec_t...	
wp2_ra_targ	wp2_ra_targ	
wp2_dec_targ	wp2_dec_targ	
sci_fov_config	sci_fov_config	
wp2_data_set_name	wp2_data_set_name	U2BX*
v3_pos_angle	v3_pos_angle	

wp2_data_set_nam	wp2_ra_targ	wp2_dec_tar(	sci_fc
U2BX0101T	5 34 29.200	+22 0 50.000	SM-1
U2BX0102T	5 34 29.200	+22 0 50.000	SM-1
U2BX0103T	5 34 29.200	+22 0 50.000	SM-1
U2BX0104T	5 34 29.200	+22 0 50.000	SM-1
U2BX0201T	5 34 29.200	+22 0 50.000	SM-1

Use navigation controls to scan results and get more data STA

Object:data

(x,y,v)=(440, 541,9189) (RA,Dec)=(5:34:40.22,22:01:11.1) (R,G,B)=(224, 66, 0)

Applet running

Figure 1: Alpha Release of STScI's archive browser StarView II. It uses ECF's utility JIPA to display a sky region from ESO's copy of the DSS catalogue together with an HST aperture overlay.

• Imposes strict adherence to object oriented design in contrast to C^{++}. In fact, this could be seen as an advantage as well.

• Programmer has no direct control over system resources. Therefore flaws of built-in services like the garbage collection algorithm are hard to circumvent.

Applet specific weaknesses

• Security restrictions make it difficult to implement basic features like saving the programme status and printing.

• Lacking browser support, latency problem due to limited bandwidth and the lack of adequate caching mechanisms can lead to frustration of users.

2. Application Briefs

DBB – The Database Browser Project

The Database Browser Project is aimed at developing a library of Java classes which can be used to create interactive database browsing and editing tools. Many data-centred applications share a common structure: A user wants to select a set of items (rows) from a database according to some searching and sorting criteria, reviews a summary of the returned items, edits one or more items and saves the edited items back to the database. The DBB classes provide the application programmer with a set of tools to quickly develop interactive tools of that kind. By specialising the library, a programmer can fine tune the look and feel of the application to the desired level. DBB is used for the new Phase II Proposal Preparation System (P2PP) V.2.

DocumentView: A way of linking GUIs and Objects

DocumentView is a set of classes that link object properties with GUI components. For instance, one can link a text field widget with the title of an Observation Block (OB). Changes in each affect the other, and thus multiple GUI components displaying the property are kept synchronised. DocumentView is a component used for the P2PP tool described below.

A New Generation Phase II Proposal Preparation Tool

The successor to the current version of P2PP (Phase II Proposal Preparation System) is an application being developed using the Java programming language. The new version is largely backwards compatible with the current system, and its basic purpose is to provide help for the preparation of Observation Blocks (OBs) both at the observer's home institution and at the telescope.

The decision to rewrite version 2 of the system in Java was based on two fundamental requirements:

• To provide easier support for multiple platforms, since Java libraries are available for most operating systems and are extremely consistent in their behaviour across platforms.

• To provide a cleaner, more intuitive user interface. One of the main aims of the Swing graphics library for Java is to provide platform-independent high-level graphical components.

Design Patterns with names like Factory and Visitor [2] were used wherever reasonable in order to reduce the learning curve for future maintainers. P2PP V.2 can function as a 2- or 3-Tier client/server system with a relational database back end and an optional business object middle tier. However, users can work completely off-line, saving unfinished work on their local machine. Connection to the ESO OB repository is only needed in order to submit their OBs for execution.

Java Interface to VLT Control Software

This class library allows Java applications to interact with the non-Java VLT Control System. It provides interfaces to the the VLT Central Common Software. These interfaces are implemented as Java Native Methods, relying on the legacy VLT Common Software libraries implemented in C/C^{++}.

The new library allows the development of high-level interfaces and VLT control applications in Java. It was made available for the first time as part of the OCT99 VLT Software Release.

P2PP V.2 uses this interface to transmit OBs to the VLT Unit Telescopes in Paranal and to get status information about their execution.

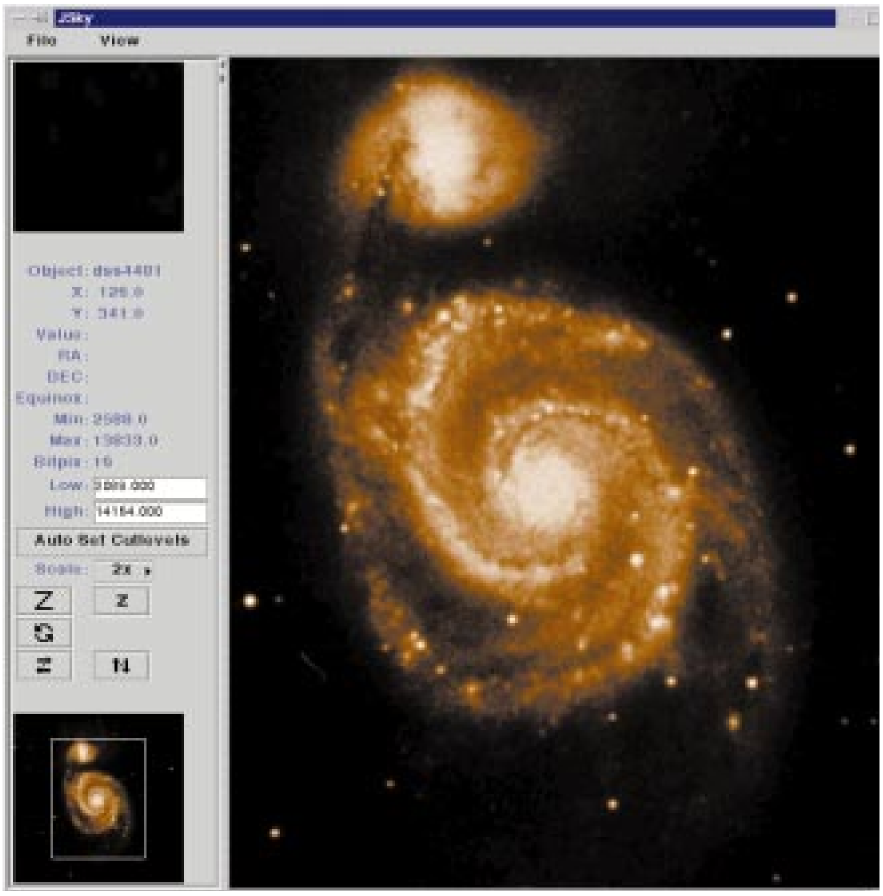


Figure 2: A first prototype of an image viewer has been developed, based on the JAI (Java Advanced Imaging) package from Sun.

Java Calibration Database Manager

The Calibration Database includes a collection of observation data and reduction procedures representing as completely as possible the observing modes and configurations of the different VLT instruments. The Calibration Database Manager is the standard application for browsing, editing, populating and aligning the local calibration database contents. This Java application is the interface between the pipeline local calibration database and the ESO archive system. It allows the ESO Quality Control Scientists of the Data Flow Operations group to request and submit files to the archive, and to structure calibration information to the formats supported by the pipeline and needed for long-term archival and trend analysis. The Calibration Database Manager and other quality control applications are described in more detail in a previous issue of *The Messenger* [3].

ESO Exposure Time Calculator

The user interface of the on-line Exposure Time Calculators provided for various VLT and NTT instrument as well as the Wide-Field Imager was implemented as a combination of HTML and Java. An observer specifies the relevant parameters in a web form, the necessary computation takes place on an ESO server and the results are returned in a page containing HTML text and a plot applet. The applet is based on Leigh Brookshaw’s GNU plot library [4].

ST-ECF Applets for the Archive Interface

ECF started its efforts in this field about four years ago. Early on applets were released on the web. They are used for previewing HST spectra and images [5]. There is JPlot – a plot utility for jitter analysis of HST observations [6]. JPlot was used for quality control purposes in first place and later on put on-line as a public archive service. There is also Skymap, a simple applet that draws locations of HST parallel observations on a map.

As a logical next step, more ambitious projects followed which are currently accomplished in collaboration with other institutions like STScI and ISO IDC [7], [8]. The aim is a pure Java interface to both HST and ESO archives (Fig. 1) that will serve as a more powerful alternative to the current web interface.

JSky – Reusable Java Components for Astronomy

The JSky development effort aims to provide a library of reusable Java components for astronomy. Based on experiences with the Skycat application, the first components being developed are for image and catalogue display and manipulation. The goal is to eventually have a collection of general-purpose and astronomy-related JavaBeans that may be easily assembled into applications using a Bean Builder, ideally with little or no hand-written code necessary.

The prototype can display FITS images and supports zooming, panning, colour maps, and automatic cut level setting. Initial tests indicate that the performance is satisfactory for small images, as long as the JAI native library (mlib) is available. At the time of writing, this is only the case under Windows and Solaris, although a Linux version is due out soon. The current version does not yet handle large images well, but can be optimised to do so. The FITS reader is based on Thomas McGlynn’s Java implementation, which currently always reads in the entire image. A future version will provide support for splitting the image into tiles, which is the only way to efficiently handle large images in JAI.

The image display application (Fig. 2) currently has a hard-coded layout. In the end, the goal is to build the application from JavaBeans and allow the users to change the layout, add new panel items, remove unwanted ones, etc.

The Java implementation of the catalogue browser looks pretty much like the Skycat version, so far, although this will likely change. The panel layout can be generated automatically, based on an eXtensible Markup Language (XML) catalogue description and the catalogue data itself may also be in XML [9]. The new catalogue interface should be more flexible than the Skycat version and allow various catalogue types to be supported by means of a catalogue registry.

JSky is an open development platform and contributions of general-purpose or astronomy-related software and components are welcome. There is a JSky home page [Table 1] as well as a JSky ftp repository [Table 1].

References

[1] A. Walsh, J. Fronchkowiak, 1998, “Java Bible”, IDG Books Worldwide Inc, p. 7–18.
[2] Gamma et al., 1995, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley.
[3] P. Ballester, A. Disaro, D. Dorigo, A. Modigliani, J.A.Pizarro de la Iglesia, “The VLT Data Quality Control System”, *The Messenger* **96**, p. 19.
[4] GNU Graph Class Library http://www.sci.usq.edu.au/staff/leighb/graph/
[5] Dolensky et al., 1998, “HST Archive Services Implemented in Java”, *The Messenger* **93**, 26.
[6] Dolensky et al., 1998, “How the Analysis of HST Engineering Telemetry Supports the WFPC2 Association Project and Enhances FOS Calibration Accuracy”, *The Messenger* **93**, 23.
[7] Binegar et al., 1999, “Browsing astronomical archives with StarView II and interacting with analysis tools such as JIPA”, in ASP Conf. Ser., *ADASS* **99**, in press.
[8] http://archive.eso.org/~mdolensk/publ/jskybof/
[9] http://vizier.u-strasbg.fr/doc/astroxml.htx
[10] TimeGraph Download Page http://www.ece.utexas.edu/~tbieleck/

E-mail: mdolensk@eso.org