TANGO APPLICATION TOOLKIT (ATK)

F. Poncet, J.L. Pons
ESRF, Grenoble, France

ABSTRACT

Tango Application Toolkit also called "ATK" is a client framework for building applications based on Java Swing in a Tango control system [1]. The goals of ATK are: to speed up the development of Tango graphical clients, to standardize the look and feel of applications and to implement the core of "any" Tango application. Tango ATK is based on Model-View-Control (MVC) design pattern. Tango ATK provides several Tango attribute viewers and Tango command viewers based on Java Swing components. In addition to these viewers, ATK provides a complete synoptic drawing and viewing system for Tango.

WHAT IS TANGO?

Tango is a CORBA based control system framework [1] [2]. It was first started at ESRF in 2000. Then the French accelerator institute SOLEIL [3] joined ESRF in 2002. Today Tango collaboration community has two additional members: ELETTRA [4] (Trieste, Italy) and ALBA [5] (Barcelona, Spain).

TANGO uses CORBA for doing network communication. The philosophy of TANGO is to provide users with an object oriented control system which is powerful and easy to use and which offers the advantages of using CORBA but hides the details. All TANGO objects are derived from one class called Device. Device is the TANGO component. It offers the following features:

- Name - every device has a name, which is unique in every instance of TANGO.
- Properties - device are configured by their properties. Properties are persistently stored items.
- Attributes - each device can have a set of data attributes. Attributes have a set of properties, which define minimum and maximum values, alarm and warning levels, event trigger levels etc.
- Commands - each device can execute a set of commands.
- Events - are asynchronous attributes, which are sent to all subscribers.

GOALS OF TANGO ATK

The main goals of ATK are the following:

- Speeding up the development of Tango graphical clients.
- Standardizing the look and feel of Tango applications.
- Implementing the core of "any" Tango application.

To achieve the first and the second goals ATK provides several swing based components to display and/or to interact with Tango device attributes and Tango device commands and also a complete synoptic viewing system. To achieve the third goal ATK takes in charge the automatic update of device data either through Tango events or by polling the device attributes. ATK takes also in charge the error handling and display. The ATK swing components are the Java Beans, so they can easily be added to a Java IDE (like NetBeans) to speed up the development of graphical control applications.

ATK ARCHITECTURE

Tango ATK is developed using the Model-View-Control [6] design pattern also used in the Java Swing package. The Tango basic objects such as device attributes and device commands provide the model for the ATK Swing based components called viewers. The models and the viewers are regrouped in two separate packages respectively ATKCore and ATKWidget.

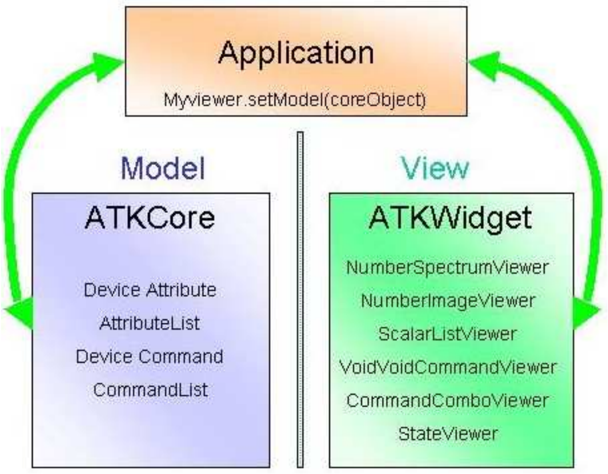
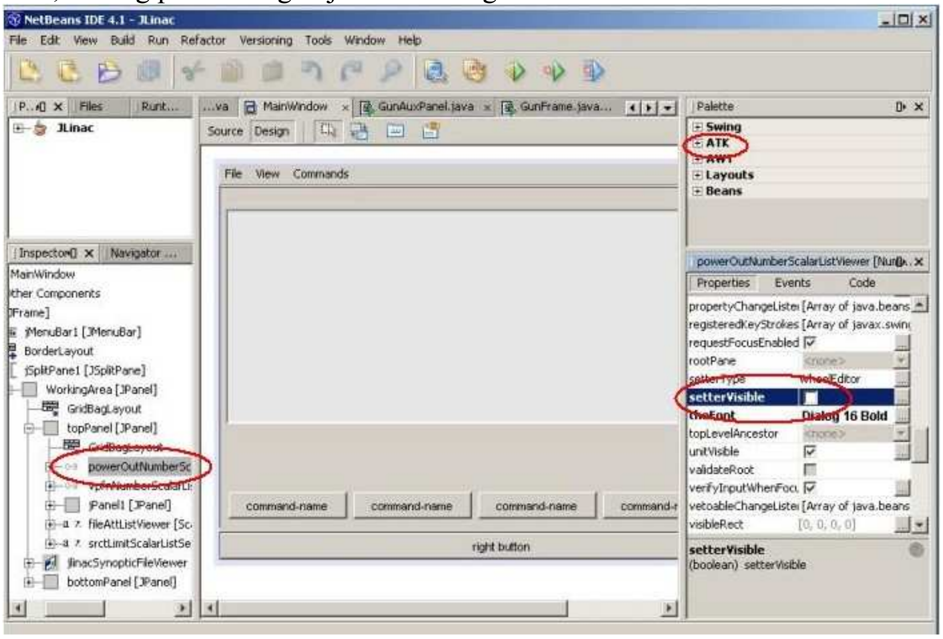


Figure 1: Model-View-Control design pattern used in Tango ATK

The application first connects to Tango device attributes and device commands and gets the reference to the "ATKcore" objects associated with them. Then the application creates the "ATKwidget" viewers. Finally the application calls the setModel method of the ATKwidget viewers in order to make the relationship between the viewer and the device attribute and/or device command to display. That is all the application needs to do. Tango ATK does the remaining work: attribute data updating and data setting, command execution and error handling.

ATK VIEWERS

ATK viewers are provided as Java Beans and as such they can easily be added in a Java based SCADA system as implemented at SOLEIL [3], or in a Java IDE (like NetBeans) to speed up the development of the graphical applications. This way the programmer can easily build up his (her) panels, mixing pure Swing objects and Tango ATK viewers.

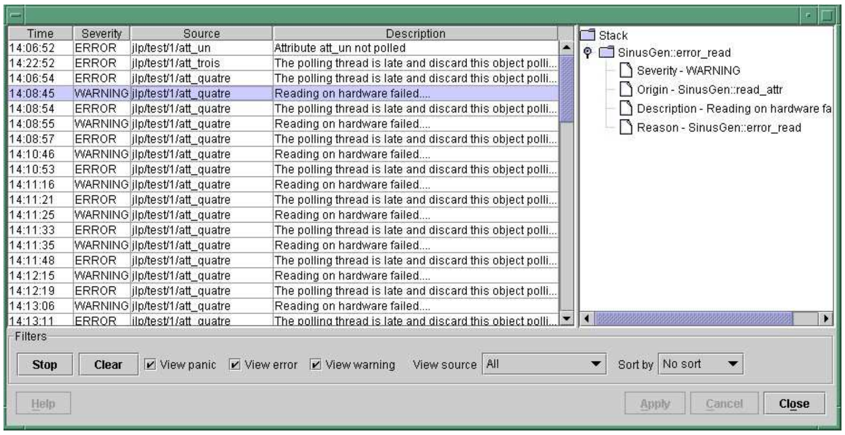


propertyChangeListe	[Array of java.beans
registeredKeyStrokes	[Array of javax.swin
requestFocusEnabled	✓
rootPane	<none>
setterType	WheelEditor
setterVisible	☐
theFont	Dialog 16 Bold
topLevelAncestor	<none>
unitVisible	✓
validateRoot	☐
verifyInputWhenFoc	✓
vetoableChangeListe	[Array of java.beans
visibleRect	[0, 0, 0, 0]

setterVisible
(boolean) setterVisible

Figure 2: NetBeans Java IDE including the ATK viewers in the Palette Manager. The bean properties of the ATK viewer are also visible in the component inspector for editing.

ATK viewers are provided for different types of Tango Components. They can be divided into different categories such as: error history window, error popup window, simple attribute viewers / editors, attribute list viewers, simple command viewers, command list viewers, …etc.



Time	Severity	Source	Description
14:06:52	ERROR	jlp/test/1/att_un	Attribute att_un not polled
14:22:52	ERROR	jlp/test/1/att_trois	The polling thread is late and discard this object polli...
14:06:54	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:08:45	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:08:54	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:08:55	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:08:57	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:10:46	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:10:53	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:11:16	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:11:21	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:11:25	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:11:33	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:11:35	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:11:48	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:12:15	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:12:19	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...
14:13:06	WARNING	jlp/test/1/att_quatre	Reading on hardware failed....
14:13:11	ERROR	jlp/test/1/att_quatre	The polling thread is late and discard this object polli...

Stack
- SinusGen::error_read
  - Severity - WARNING
  - Origin - SinusGen::read_attr
  - Description - Reading on hardware fa
  - Reason - SinusGen::error_read

Filters: ✓ View panic ✓ View error ✓ View warning View source All Sort by No sort

Figure 3: ATK Error History viewer

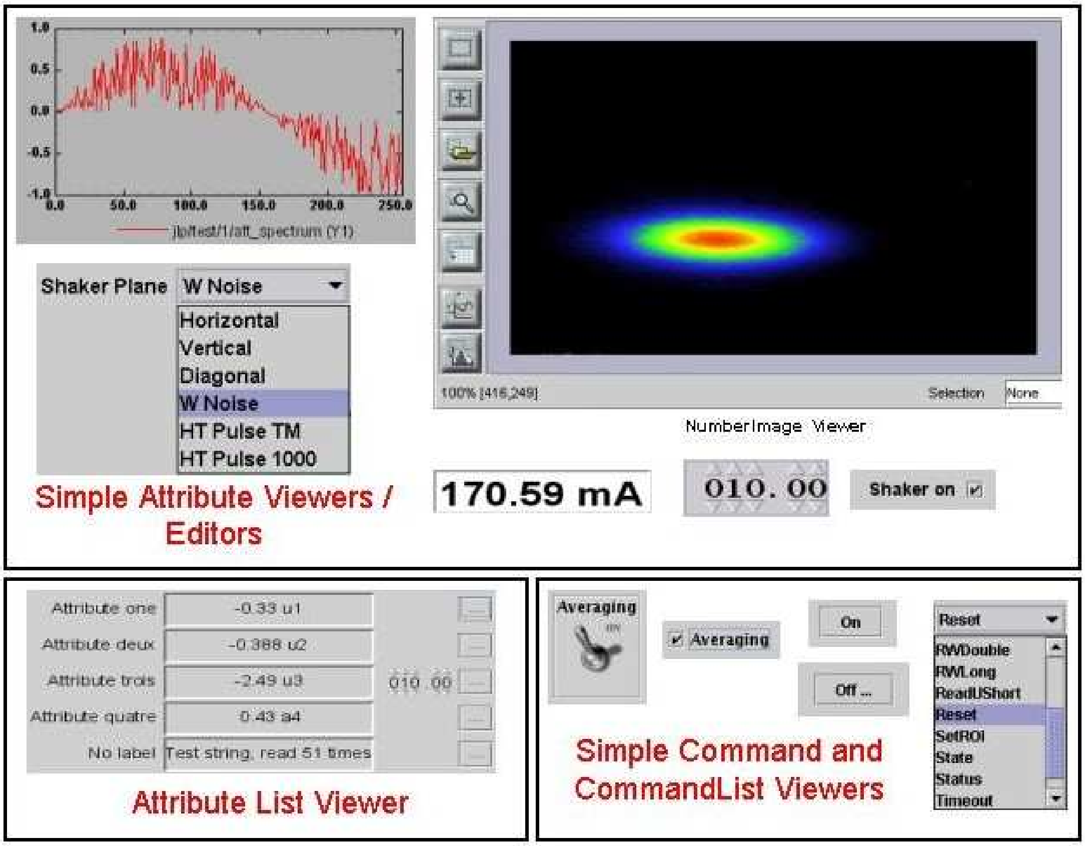


Figure 4: ATK attribute and command viewers

SYNOPTIC DRAWING AND VIEWING

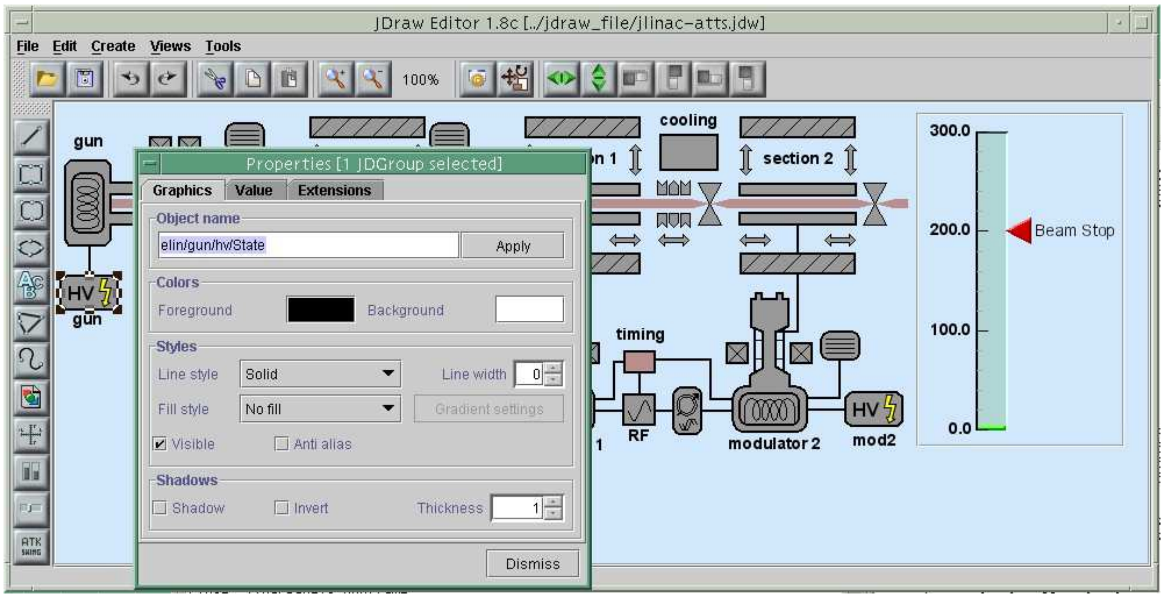
A synoptic is a drawing in which each object can be linked to a Tango object. A part of the synoptic drawing can be linked to the state attribute of a Tango device where another part is associated to a numerical attribute of another Tango device. The main idea of synoptic drawing and viewing system is to provide the application designer with a simple way to draw a synoptic and to animate it at runtime according to the values and states read from the control system. ATK provides two components for this purpose:

- A graphical editor called “Jdraw”
- A synoptic viewer: SynopticFileViewer

In the first stage the application designer draws the synoptic using the graphical editor “Jdraw” provided within ATK. During this stage, the application designer associates to each part of the drawing a Tango object name (device attribute, device command) respecting the Tango naming convention. The names given at the drawing stage will be used by ATK at run time to connect to the corresponding Tango objects in the control system. The parts of the drawing with no “Tango” name will not be animated at run time.

The application designer can also associate to each named part of the drawing a fully defined java class name. This java class will be instantiated at runtime by ATK when the user clicks on this part of the synoptic drawing. This way the application designer can associate to synoptic objects, specific panels he (she) wants to be popped up when the object is selected.

The synoptic drawing is then saved to an ASCII file. The file format is specific to ATK Jdraw.


JDraw Editor 1.8c [../jdraw_file/jlinac-atts.jdw]

Properties [1 JDGroup selected]

Object name: elin/gun/hv/State

Line style: Solid
Line width: 0
Fill style: No fill
[✓ Visible] Anti alias
Shadow, Invert: unchecked
Thickness: 1

Figure 5: ATK Jdraw Editor

Once the synoptic drawing is saved to a file, the application programmer can build the application’s window using a Java IDE (NetBeans) and putting inside the “ATKwidget” synoptic file viewer. The name of the synoptic drawing file is passed through the corresponding java bean property. This is all you need to do for viewing an animated synoptic using Tango ATK.

At runtime the synoptic viewer parses and browses the synoptic drawing to discover which Tango objects are referenced inside the file. ATK tries to connect to all these Tango objects (attributes, commands) and if the connection is successful then the corresponding drawing object will be animated according to the value of Tango object. The animation depends on the attribute type. For example the default behavior for a state attribute is to change the background color of the drawing object according to the state value.

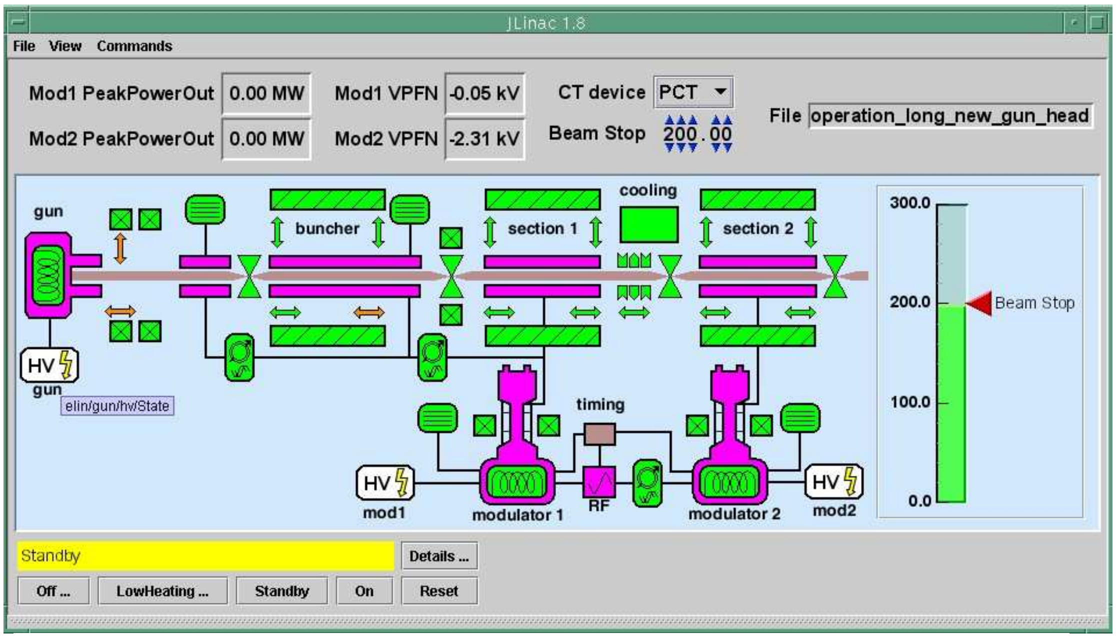


Figure 6: The synoptic integrated in an application window and animated at run time

CONCLUSION

Control systems need several separate client applications to interface the hardware. Development of reliable and user friendly client software is a time consuming process. Moreover the same problems are solved in a different manner in each application code. Tango ATK intends to solve the common Tango client programming issues in one well tested place. This way, ATK provides reusable code for the development of the client applications and contributes to the their reliability. Using the ATK components does not force the application designer to use a specific application framework because they can easily be added as Java Beans in any Java application builder or framework.

REFERENCES

[1] http://www.esrf.fr/tango - the TANGO website
[2] “TANGO a CORBA based Control System”, ICALEPCS’2003, Gyeongiu, Korea, October 2003.
[3] http://www-controle.synchrotron-soleil.fr:8001/collaboration
[4] http://www.elettra.trieste.it/~tango/index.html
[5] http://www.cells.es/
[6] “Design Patterns: Elements of Reusable Object-Oriented Software” by E.Gamma, R.Helm, R.Johnson, Addison-Wesley, 1995