Tcl/Tk based Framework – A Lynchpin in Development of Instruments for Remote Sensing

Authors

Amit Dave (amitdave@sac.isro.gov.in), Jitendra Sharma, Ashutosh Dutt, Anil Sukheja, Ashish Mishra and D.R.Goswami

PCEG/SEDA, Space Applications Centre (ISRO), Ahmedabad-380015, India

Abstract

Space Applications Centre, ISRO develops Electro-Optical sensors for Indian Remote Sensing (IRS) program, Inter-planetary missions and airborne imaging campaigns. The sensors are of various types, complexities and their development has a typical develop-test-use cycle. A centralized system for Characterization and Performance parameter Evaluation of sensors ï named XSCoPE, provides an end to end solution for data acquisition, parametric evaluation, visualizations and archival.

A set of in-house developed tools called 'Arsenal' meet the parametric evaluation requirements. A pure Tcl shell around the Arsenal tools, called Arsenal Shell (ASH!) glues them with the standard Unix tools to meet the specific requirements. An ASH!Server has been built using Tcl on Linux based systems to facilitate communication with the client applications. Tcl/Tk has been chosen because of its simplicity in building the server applications, package based architecture, gluing, error handling, GUI and output processing. With the help of simplified data structures of Tcl, the server could be designed in such a way that the cluster of similar servers, meet the load balancing requirements to support multiple and simultaneous sensor development. Sensor specific requirements and common requirements are met with the help of 'macros' which provide abstraction.

The paper describes how various Tcl features have been applied to realize complex scientific software applications, exhaustively used in various sensor development programs such as ISROôs Mars Orbiter Mission (MOM) to meet the overall objectives of sensor development.

Keywords

Remote Sensing, ISRO, performance parameters, Tickle, Widgets, GUI, network, Tcl/Tk

[1] Introduction to XSCoPE

The sensors for space-borne, air-borne and inter planetary science applications pose different developmental challenges. Their development life cycle includes conceptualization, realization, testing and optimization. For different category of sensors, the requirements differ in terms of following activities:

Å Data Acquisition: Interfaces and data rates

Å Data architecture

Å Parametric evaluation, algorithms and optimizations

Å Visualizations

Figure-1 shows the centralized system for characterization and performance evaluation, XSCoPE.

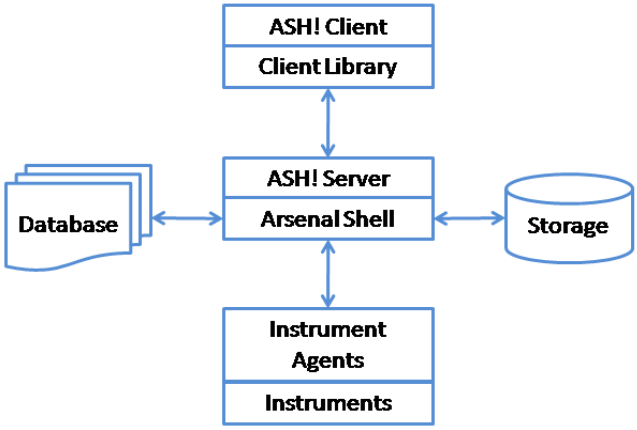

Figure-1: XSCoPE System

The XSCoPE system supports these developmental activities from the bread board level to the flight model of the sensors. It computes sensor performance parameters like Signal to Noise Ratio (SNR), Modulation Transfer Function (MTF), Band-to-Band Registration (BBR), Distortion, Spectral Response, Radiometric calibration coefficients etc. Apart from the above listed requirements, following activities are also required to be carried out during various development phases of a sensor:

Å Data Archival;

Å Making test results, reports and data available to users;

Å Keeping session data up to date on all servers in cluster environment;

Å Software configuration and system management.

To meet these requirements the XSCoPE system is designed with a layered approach as discussed below and shown in Figure-2.

[2] Layered Architecture of XSCoPE

A. Layer-0 Arsenal Tools

A set of in-house developed tools called áArsenalôforms the bottom most layer. It is meant for number crunching, data analysis and is optimized for performance and actually deals with the data to perform various operations like data acquisition, data realignment, rearrangement, computation of various parameters, formatting etc.

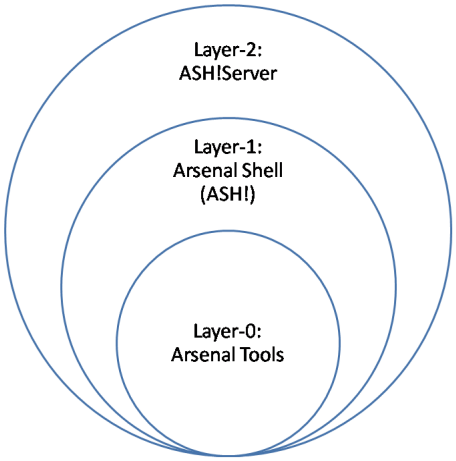

Figure-2: Layered Structure of XSCoPE

B. Layer-1 Arsenal Shell

Outer layer ASH! ï a pure Tcl shell ï is a collection of packages. Each package meets one or more requirements by gluing Arsenal tools and standard Unix commands together. A package represents a Tcl command and supports multiple options in standard Tcl syntax. Certain common tasks applicable to all packages are handled by the abstraction mechanism, implemented by overriding *procs*. Object oriented approach makes an abstract package that provides option processing, help, entry point, execution and cleanup mechanism. Options and internal variables of the packages have been encapsulated using their private namespaces.

C. Layer-2 ASH!Server

Outermost layer is called an 'ASH!Server' built on top of the shell to extend the shell features, by facilitating communication with the client applications. Clients provide a graphical interface and invoke a series of commands for performing various end-user tasks.

With this framework, an application to meet a new requirement can be easily developed. Tcl/Tk has been used in building Layer-1 and Layer-2; as the nature of the software is such. Tcl allows building large applications using package based architecture, namespaces and error handling.

[3] Applications of Tcl/Tk Features

This section lists the Tcl/Tk features, which helped in developing various system components.

A. Gluing

Tclôs ósocô command and its options allow invoking Arsenal tool and other Unix commands, opening command pipeline, reading output and status from the pipe. Moreover, the command output can be re-directed to different channels. This gluing feature is useful in various packages.

B. Packages and Namespaces

All ASH! commands have been encapsulated in different packages and it is the smallest entity, which can be individually developed, tested and maintained. Packages have their own namespaces to keep variables and avoid cluttering. This is essential for the large software projects during development and maintenance. Moreover, being an interpreted language, packages can be re-loaded in the running server without shutting it down. Package versioning ensures the appropriate and up-to-date revision of the source code on all servers.

With the extension capability [1][2] using the C/C++ code, an ópcôpackage has been developed to support inter-process communication. It supports semaphore, message queue and shared memory using standard POSIX calls.

C. Error handling

Tclôs error handling mechanism, which provides *error* command and associated global variables viz. the *errorCode* and *errorInfo*, is useful for generating error stack while delivering the error. This makes the end product robust and almost ócrash-proofô

D. Abstraction and Overriding

Object oriented features provide abstraction to the packages for standardizing certain common features across the packages. The command option processing, initialization, cleanup, help and installation-check procedures have been developed using this feature.

E. Data Structures

Standard data structures supported by Tcl viz. the strings, lists and associative-arrays have been used throughout the software. Over and above the basic structures, those provided by ódictôand ódomôhave been used for registry and larger

datasets, which has enabled in-memory data representation for holding client session data.

F. Communication over network

Client-server communication has been built using 'socket' for implementing communication mechanism. ‗httpô provides a way of representing ASH! objects as standard URLs and is used for communication with the web server.

G. Widgets

With Tk, GUI applications can be built very fast and with the help of BWidget package support, almost all GUI controls are possible. A GUI for the test application for the ASH!Server has been developed using the standard Tk and BWidget controls.

The application uses Canvas widget to produce an icon based view of the data objects on the server. Tree widget is used to present another view for navigation and associate callback actions with the data objects. The ‗**ASH!>**ô prompt and console has been created with the Text widget.

H. Image display

Tk provides in-built support for bitmaps and GIF file formats. With other extensions, it is possible to display popular file formats like JPG and PNG. With this support the sensor data can be presented as an image in an application. This has been used extensively in the client applications built using Tk.

I. Cross platform support

The client application called ASH!Client is targeted to run on Linux as well as Windows. The Tcl/Tk cross-platform feature provides mechanisms to configure application for different platforms. With the help of ‗AndroWISHô the application could be migrated to Android tablet without any change.

J. Database support

‗oratclô package has been used to interface with the Oracle database and methods are established for database assisted features.

K. Un-manned system

‗expectô extension [3] has made the system administration ï an unmanned job. Couple of ‗expectô scripts combined with Linuxôs rsync is powerful enough to archive sensor data on storage and syncing the contents in participating cluster servers.

To summarize, following advantages have been derived using Tcl/Tk features:

- A protocol [4] has been devised over TCP/IP for controlling the lab instruments like Micro-positioner, Mono-chromator, Power Supply, Digital Multi-Meter etc. Controlling agents have been developed to establish communication between

XSCoPE and instruments using interfaces like RS-232, USB, GPIB and Ethernet network. Automation [5] scripts have been built for certain test benches requiring repeated tasks involving human intervention.

- A server has been designed in such a way that the cluster of similar servers make an ASH!Cluster. The cluster meets the load balancing requirements for supporting multiple and simultaneous sensor developments.
- Series of ASH! extension commands put together with other Tcl control structures, a set of macros have been devised. With the help of namespaces, they are classified as global or local that provides abstraction to enable sensor specific processing.
- With ease of learning [1][6][7], Tcl allows reduced application development time in response to a new requirement. The entire application has been built with version 8.4 which is available just out of the Linux box. Additional packages have been made part of the installation with ôauto_pathõsupport.
- A lot of Tcl scripts have been developed to perform the offline data analysis tasks to cater to various project requirements.

[4] ASH! Packages

The packages listed in Table-1 are part of the Arsenal Shell as of now. The packages fall in different classes as per their application. These classes are:

1. Data acquisition
2. Pre-processing
3. Parametric evaluation
4. Offline data analysis
5. Visualizations
6. Results generation
7. Data archival
8. System management and monitoring

Table-1: Classification of ASH! Packages

Class	**Packages**
Data acquisition	acquire, icp, satinfo
Pre-processing	bpc, dark, filter, split
Parametric evaluation	ashexpr, bbr, bitops, calana, chi, compu, fft, ltc, psf, qstat, radc, snr, srm, stagc, swrc
Offline data analysis	extract, header, selection
Visualizations	img, imgproc, plot, splot, view
Results generation	convert, xresult
Data archival	dbase, store
System management and monitoring	bg, commonlib, context, help, macro, misc, project, system, vars, webif

[5] Current Applications

A. ASH!> Test bed

For testing and verification of the package functionality, a test bed called a ASH!Client has been developed. It offers an **ASH!>** prompt, command output area, a visualization for server side objects, command history and other GUI based actions, as shown in Figure-3.

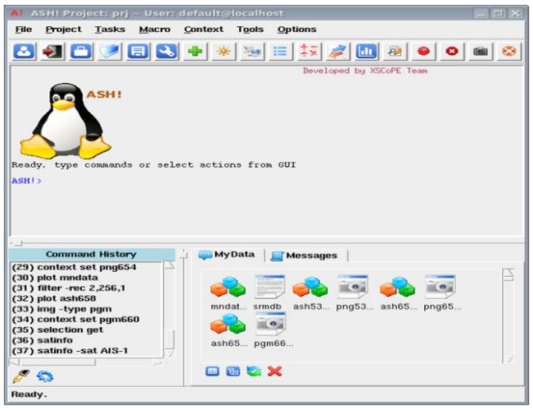

Figure-3: ASH!Client using Tk

B. Application for Evaluating MOM Instruments

Figure-4 shows the test bench application developed for performance evaluation of MOM sensors.

Figure-4: MOM Test and Evaluation Application

C. Example Macro

A typical macro with ASH! commands put together, is shown in Figure-5 as an example. The macro acquires data for LISS3 multispectral camera for three spectral bands and produces objects like data and plots.

```
foreach band { B2 B3 B4 } {
 icp –inst SIS –exec “CONFIG-MAIN”

 acquire –sat RS2A –payload LISS3 –band $band \
   –variable acqdata-$band

 qstat -op xres –heading B2,B3,B4 –header \
   –variable Statistics-$band

 context set acqdata-$band:1

 plot –png –title “Mean Data Profile - $band” \
   –labels  “Pixel,DN” \
   –variable plot-$band
}
```

Figure-5: Typical Example Macro ï A Language of Checkout

[6] Future Applications

1. With AndroWISH, the Tcl/Tk code can be migrated to the Android tablets. This will allow limited area mobility around the test bench in laboratories for instrument interface.
2. With JSON support, the ASH! client-server interface can be greatly simplified.
3. TclHttpd [8] provides a way of extending the applications through web to ease the client-server interface.
4. With Tclôs very small memory foot-print, the interpreter can be embedded in an on-board [9] hardware/computer. This is an ideal solution for the platforms like Landers and Rovers for planetary exploration missions.
5. Using the extensive database support, work flow requirement can be met to ease the simultaneous sensor development activities.

[7] Conclusion

ASH! project is developed in-line with the Tcl philosophy of extensibility. Packages and macros provide the extension capability to ASH! to make it amenable for continuous evolution for the future sensor development requirements.

With the layered structure, the computational needs, performance requirements, system related requirements and development responsibilities are handled at different layers. Tcl ôconnectsô or ôstitchesôthe layers together.

So far, the clients have been developed in Tk, PHP, Java and LabVIEW. Application porting on Android for different applications has also been explored. This proves the strength of Tcl in designing framework. To meet the overall sensors checkout and evaluation requirements, the framework plays a crucial central role.

Acknowledgements

The authors would like to thank to Shri A.S. Kiran Kumar, Director, SAC-ISRO for providing opportunity to work for the development of Checkout and Evaluation

software for all IRS and inter-planetary sensors and providing valuable inputs time to time. We sincerely acknowledge constant guidance and encouragement by Shri. Saji A. Kuriakose, Deputy Director, SEDA.

References

[1] Clif Flynt, ñ*Tcl/Tk – A Developer's Guide*ò, 2nd Edition

[2] D.J. Asson, A. Bose, A. Krueger, ñ*A Tcl/Tk-Based, Intelligent Graphical Editor for Preparing HST Programs*ò, Astronomical Data Analysis Software and Systems V, ASP Conference Series, Vol. 101, 1996

[3] Don Libes, ñ*Exploring Expect*ò, O'Reilly Inc., ISBN 1-56592-090-2

[4] Amit Dave, Jitendra Sharma, Ashutosh Dutt and Anil Sukheja, ñ*Generic protocol for seamless control of test instrumentation towards realization of electro-optical sensors*ò, IEEE Recent Advances in Intelligent Computational Systems, Sep 2011.

[5] *Design Space Explorer*, Quartus II FPGA DesignTool by Altera

[6] John K. Ousterhout, ñ*Tcl and the Tk Toolkit*ò, Addison-Wesley, ISBN 0-201-63337-X

[7] Web portal www.tcl.tk online reference.

[8] Brent Welch, ñ*The TclHttpd Web Server*ò, Scriptics Corporation

[9] David E. Smyth, ñ*Tcl and Concurrent Object-Oriented Flight Software: Tcl on Mars*ò, Mars Pathfinder Flight Software Team, JPL/NASA