Development of Software Interfaces using TCL/Tk

Authors

Amit Dave (amitdave@sac.isro.gov.in), Jitendra Sharma, Anil Sukheja, Sumit Kumar, Nutan Kumari, Heena and Parth Nakum

Space Applications Centre (ISRO), Ahmedabad-380015, India

Abstract

Interfaces are essential elements of a complex software system and one of the key aspect of system engineering practices. Well defined soft interfaces decide the health of a system, makes them rugged enough to withstand changes. Programming language selected, for the system under development, should have Application Programmer's Interfaces (API) to clearly define the interfaces. More importantly the language API should make interface development easier for building a complex system. Toolkit Command Language (TCL/Tk) provides simple mechanisms for building the server applications, package based architecture, gluing, error handling, GUI and output processing.[1][2] TCL has packages for processing data objects using XML, JSON, SOAP and transport using HTTP. Space Applications Centre (SAC), ISRO develops electro-optical sensors for its remote sensing programme. These sensors are exhaustively tested for their performance during development, using a software system called XSCoPE – A System for EO Payload Evaluation. The system can handle different types of sensors concurrently in different stages of their development. One of the layer of XSCoPE, called ASH!Server is entirely written using TCL and uses different software interface methods. This paper discusses various information interface mechanisms and describes them using TCL, in the context of a system like XSCoPE.

Keywords

Remote Sensing, ISRO, JSON, XML, performance parameters, software engineering, TCL/Tk, API

[1] Introduction to XSCoPE

XSCoPE – A Linux based Evaluation System [3] for Electro-Optical payloads is one of the key elements for development of sensors for various earth observation and planetary missions by ISRO. The system is being used for all types of sensors and its architecture is depicted in Figure 1.

The central block – called Arsenal Shell (ASH!) – is a collection of TCL packages for various tasks. Its services are exported using a component called ASH!Server on top of the shell.

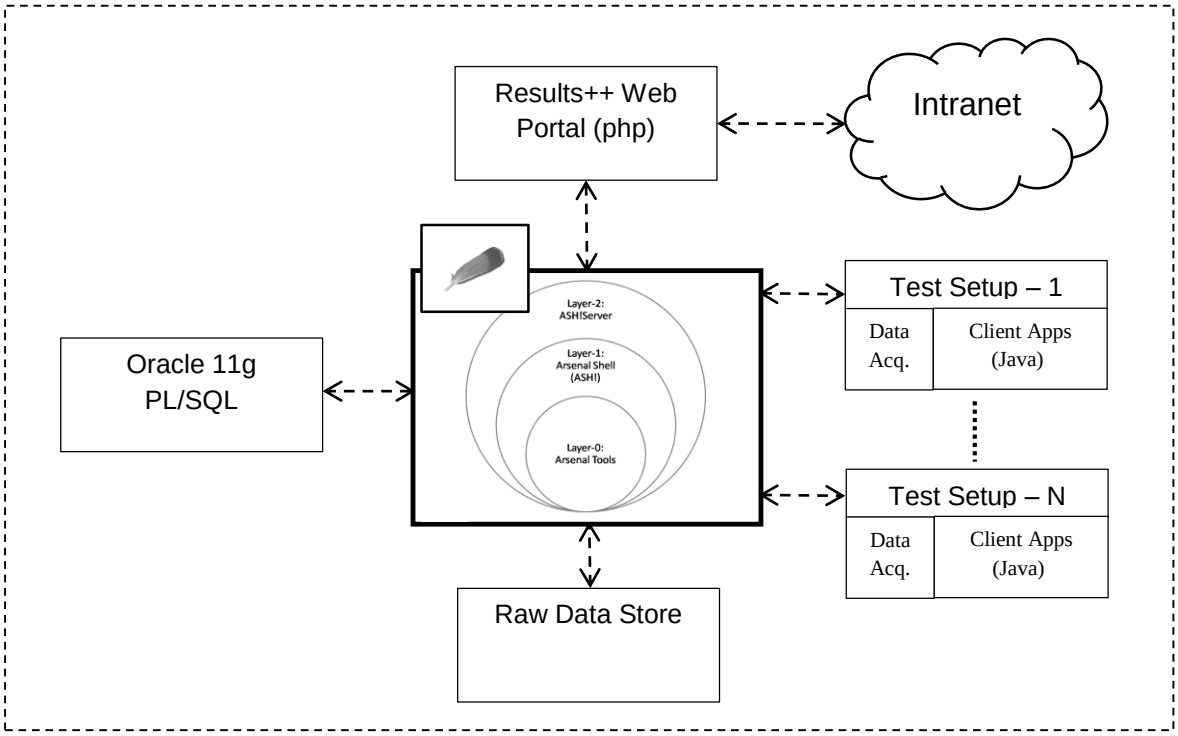

Figure-1: XSCoPE Architecture

As seen in Figure-1, the overall system requirements are met by various sub-systems, which use different software and '*speak'* their native languages. Therefore, the need of interfacing between these sub-systems arises and this is important in overall architecture. This section provides functions and brief description of the sub-systems.

The test setup(s) comprises:

(1) data acquisition system

(2) sensor commanding and instrument control apps

(3) client applications for individual test bench.

Data acquisition system has hardware and software elements customized for the sensor under test. It deals with the data rates and pre-processing complexities of the sensor. Commanding module generates necessary interface signals for the sensor. Instrument control apps provides interfaces with the peripheral hardware like power supply, micro-positioner, light source etc. Client application provides front-end for user interaction and visualizations for various test bench specific tasks.

The central block of the system called ASH!Server provides its services to the client applications over a high-speed 10G intranet. It provides range of services as follows:

- raw data acquisition,
- pre-processing of data and restructuring,
- computation of performance parameters,
- visualizations generation,
- image generation and pre-processing,
- data and test results archival,
- abstraction of database tasks,
- offline data analysis,
- macro handling,
- multi-client support,
- user management and clustering

These functions are divided into ASH! Packages, with each package handling one or more tasks by invoking Arsenal tools and/or Linux utilities from the bottom layer.

Arsenal layer is the bottom most layer, which is formed by a set of in-house tools developed in C/C++. They provide various functions for computing performance parameters, data manipulation, formatting etc. These tools, combined with standard Linux utilities, meet the majority of requirements. For advanced/complex computation needs, MATLAB executables can be made and invoked.

The sub-system interconnections are made up of standard interfaces as described in the subsequent sections.

[2] Methods to exchange data between sub-systems

Various methods exist to exchange data between the software sub-systems, few of which are discussed here.

XML (eXtensible Markup Language) is one of the most widely used methods of communication between the software elements. Combined with SOAP and RPC it becomes a powerful technique to create rugged software environment. Many software applications use XML base to create their own language to establish communication between server applications and agent software. Google's Keyhole Markup Language (KML), Geography Markup Language by Open Geospatial Consortium and Scalable Vector Graphics (SVG) by W3C are popular XML dialects to name a few.

JSON (JavaScript Object Notation) is a lightweight, text based, language independent data interchange format and is more popular in Web environment. The JSON format is used to describe objects and exchange them over a network connection between a server and web application, serving as an alternative to XML. Both JSON and XML

are used for serializing objects but JSON being more verbose, it is efficient over network.

In the CSV (Comma Separated Values), TSV (Tab Separated Values) or simply delimited values, information to be exchanged is delimited with comma, tab or some such character, agreed upon by both the software. When amount of data to be exchanged is large, XML or JSON being more verbose, their parsing needs more resources. In such situations, CSV is most suitable. Its disadvantage is that, it makes a *hardwired custom* interface between the software elements and it becomes difficult to handle changes on both ends.

For large data objects, binary data formats provide even more efficient way of information exchange. Packetized data objects representations as defined by CCSDS[4] standard are popular amongst the data links based communication and applications. Tele-communication instruments and imaging sensor hardware typically generate such standardized streams of data. However, these formats make the systems rigid and not so amenable to changes.

Interface mechanisms are selected considering the factors like amount of data, nature of software sub-systems and application.

[3] XSCoPE Interfaces

XSCoPE sub-systems are shown in Figure-1. Various interfaces mechanisms used between the sub-systems are discussed in this section.

ASH!Server and Data Acquisition System

Video data acquisition source generates binary data streams, whose format varies across sensors and hence, it is in a non-standard form. The data formats are generally dictated by the hardware.

ASH!Server and Test Setup Client Apps

A sensor test setup block comprises the following:

1. data acquisition system (video and telemetry), which in turn connects to a sensor under test;
2. a client application (ExpressClient) on which visualizations appear;
3. instrument agent software for laboratory peripheral instruments like micro-positioner, light/spectral source, power supply etc. [5]

ExpressClient application is developed using Java and it '*talks'* to ASH!Server by invoking commands. This is a plain text information, where the information passed to the server, is in TCL's *cmdline* syntax. The information returned by the server, in response to the command execution, is in various forms and depends on the command invoked.

For data plotting purpose, generated data is converted to delimited values (in this case CSV) and used by plotting utility in client for generating interactive plots.

Sensor performance parameters viz. signal-to-noise ratio, square wave response, band to band registration, along with other meta-data are represented as XML objects in XResult markup format as shown in Figure-2. *XResult* objects when retrieved, are formatted and displayed using Java's *JTable* control and in *XSLT* when displayed in a web browser.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="xres.xsl" ?>
<XResultTable>
    <Header>
        <Satellite value="RS2A"/>
        <Payload value="LISS4"/>
        <TestName value="BBR@PTNE"/>
        <Condition value="BBR"/>
        <AcqMode value="IMAGE"/>
        <AcqDataMode value="RAW+AVG"/>
        <AcqDate value="20-Apr-2016"/>
        <AcqTime value="19:51:55"/>
        <AcqSpxl value="1"/>
        <AcqNpxl value="12000"/>
        <AcqNBands value="3"/>
```

Figure-2: XResult markup

ASH!Server and database

Oracle 11g database server archives all test results, generated by the sensors. Test results are formatted in *XResult* markup format and stored in Oracle 11g database for archival purpose. A set of PL/SQL procedures handle the *XResult* object.

ASH!Server and Results++ Intranet Portal

Test results and raw data archive are accessible through an intranet portal called *Results*++, which also facilitates raw data analysis and downloads. Results++ portal invokes TCL macros on ASH!Server, which produce data objects and their representation in JSON format. Stored results in *XResult* format are rendered using XSLT stylesheets on a web page.

Table-1 summarizes interfaces between the XSCoPE subsystems:

Table-1: Summary of Interfaces

Sub-Systems Involved		Interface (Type)
ASH!Server	Data Acquisition System	Binary (non-standard)
ASH!Server	ExpressClient Applications	XResult, CSV (standard)
ASH!Server	Oracle 11g	XResult – an XML dialect (standard)

ASH!Server	Results++ Portal	XResult – an XML dialect, JSON (standard)

[4] Interface Mechanisms provided by TCL

This section describes the TCL mechanisms in the context of utilization in XSCoPE software system.

XML

TCL provides extensions for parsing XML contents. It offers (1) SAX – stream oriented parsing and (2) DOM – document oriented parsing. tDOM and TCLXML/TCLDOM are the two main TCL extensions providing parsers[8]. XSCoPE uses tDOM package by creating document objects, as the need is such.

JSON

TCLlib 'json' package provides JSON parser and generator. Figure-3 shows a typical JSON objects.

```
{
"op_objects" :
  [
   { "context_name"  : "meanPlot" ,     "context_type"  : "png" ,  "context_title" : "Mean Plot"    } ,
   { "context_name"  : "sdPlot" ,       "context_type"  : "png" ,  "context_title" : "SDEV Plot"    } ,
   { "context_name"  : "stat" ,         "context_type"  : "html" , "context_title" : "Statistics"  } ,
   { "context_name"  : "hkdata" ,       "context_type"  : "html" , "context_title" : "HK Data"  }
  ]
}
```

Figure-3: A typical JSON object used in XSCoPE

Most languages require that JSON be first converted to a native representation, and later serialized back to a string, while TCL provides the means (through the TCL_Obj mechanism) to efficiently manipulate the JSON directly [6].

CSV

'csv' package in TCLlib provides mechanisms for dealing with CSV multi-line data. It allows converting CSV to HTML, cutting CSV columns, joining two CSV data sets and sorting[8]. ASH!Server produces certain data in this form, which are used for visualizations, downloading, exporting to Microsoft Excel, Matlab or such data analysis tools or importing into Oracle11g.

[5] Conclusion

Interfaces between sub-systems play an important role in building a complex software system. Standard data exchange methodologies should be supported by the language being used so that sub-systems can be designed, developed and maintained comfortably. TCL provides industry standard mechanisms for developing software interfaces. TCL itself being a lightweight scripting language, interface mechanisms also exhibit similar characteristics. With emerging interface technologies that can

quickly be embedded into standard TCL distribution or as third party packages, it proves to be an ideal platform for building reliable complex systems [7].

[6] Future Scope

TclSOAP, a Tcl package, provides method binding for Tcl clients to remote procedures and implemented using either SOAP, XML-RPC or JSON-RPC [8]. They use XML and JSON for formatting the data and use HTTP for transport to and from the service provider. Combined with 'tclhttpd' and RPC, servers can be built. In future, certain mechanisms within XSCoPE shall be replaced with these concepts to achieve more robustness.

Acknowledgements

We sincerely acknowledge constant motivation and encouragement provided by Mr. Ashish Mishra, Division head, and Mr. D.R. Goswami, Group Director, Payload Checkout Electronics Group. We also thank Mr. S.S. Sarkar, Deputy Director, SEDA for providing the opportunity to work on Payload Evaluation System.

References

[1] Clif Flynt, "*TCL/Tk – A Developer's Guide*", 2nd Edition

[2] John K. Ousterhout, "*TCL and the Tk Toolkit*", Addison-Wesley, ISBN 0-201-63337-X

[3] Amit Dave, Jitendra Sharma, Ashutosh Dutt, Anil Sukheja, Ashish Mishra and D.R.Goswami, "*TCL/Tk based Framework – A Lynchpin in Development of Instruments for Remote Sensing"*, 21st TCL/Tk Conference November 2014, USA.

[4] CCSDS 133.0-B-1 BLUE BOOK for Space Packet Protocol, September 2003

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

[6] Cyan Ogilvie, Ruby Lane, Inc. "JSON as a Native TCL Value", October 24, 2016

[7] Zach Conn, FlightAware, "Hyperfeed: FlightAware's parallel flight tracking engine"

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