SWIMming in Tcl

Frank Morlang
German Aerospace Center (DLR)
Lilienthalplatz 7
38108 Braunschweig, Germany
frank.morlang@dlr.de

ABSTRACT

Against the background that all future air traffic participants are requested to act as System Wide Information Management (SWIM) communicating sub-systems by the future Single European Sky Air Traffic Management Research (SESAR) SWIM 'Intranet for ATM' concept, SWIM compliance of future ATM information sharing software modules is of utmost importance. The paper describes Tcl/Tk usage in this context, focusing on a prototype referring future space / air traffic integration needs.

1. INTRODUCTION

The SWIM concept can be summarized to its role of providing the right data to the right target(s) at the right time, associated by the connectivity between the right sources and sinks. Strictly obeying to this elemental rule, generates the need that future commercial spacecraft will have fulfil SWIM compliancy, especially against the background of the seamless integration into normal air traffic and the generation and distribution of emergency information in case of a major hazard event [1][2].

2. PROTOTYPE

2.1 General Architecture

The prototype's general architecture is shown in Figure 1. It is realized as a Hypertext Transfer Protocol Secure (HTTPS) web server, containing a provisional hazard area model which assumes a Space Shuttle like trajectory of the hypothetical spacecraft. The model performs inter-/extrapolation based on the United States National Aeronautics and Space Administration's (NASA) Columbia space shuttle accident debris data [3].

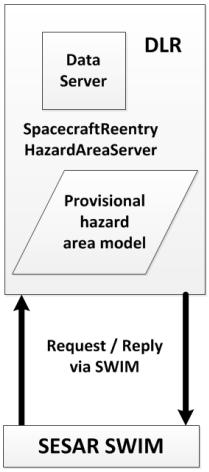

Figure 1. General architecture of the SpacecraftReentry-HazardAreaServer

2.2 SESAR SWIM / FAA SWIM

The Europe / U.S. SWIM harmonization architecture is presented in Figure 2. It demonstrated the connection between SESAR SWIM and the United States Federal Aviation Administration's (FAA) SWIM concept of Global Enterprise Messaging Service (GEMS), where Service by NEAR (SBN) was one representational implementation of a GEMS provided by the Next Generation ERAU Applied Research (NEAR) lab. The demonstration use case included Flight Object data exchange using SWIM's data interchange standard Flight Information Exchange Model (FIXM) version 3.0.1 between an FAA research and development Flight Object Exchange Service, an airline Electronic Flight Bag (EFB), and the SpacecraftReentryHazardAreaService, consuming the aircraft state data within the Flight Object FIXM message and published its output using SWIM's data interchange standard Aeronautical Information Exchange Model (AIXM) version 5.1 for consumption and display by other interested parties.

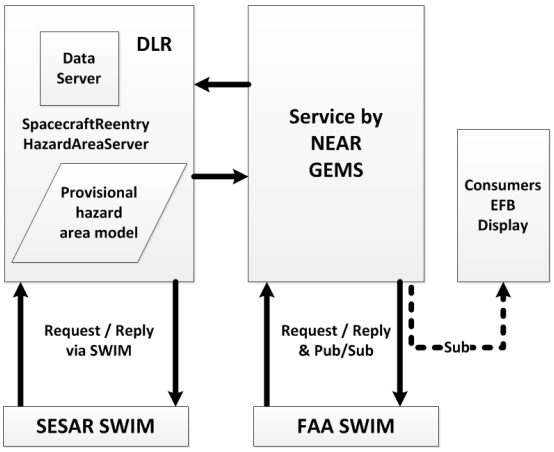

Figure 2. SESAR SWIM / FAA SWIM demonstration architecture

3. REALIZATION

The SpacecraftReentryHazardAreaServer prototype is developed in 100 % Tcl, benefiting from the following packages:

- TclHttpd as the web server,
- Web Services for Tcl for the server side web service creation on top of TclHttpd,
- TclTLS for using HTTPS,
- Rpcvar for complex data type definitions,
- CriTcl for improved performance by the usage of C code runtime embedding

- BaseXClient-Tcl for using the BaseX server protocol to communicate with the hazard area model database server.

Five Tcl modules build the prototype's structure (Figure 3). The SpacecraftReentryHazardAreaServer.tcl main module represents the generic start component, setting up the general webserver frame and loading the hazard area model database. Defineweb-services.tcl is just a mediator for sourcing services' code files, only one in the prototype's case, the SpacecraftReentryHaz-ardAreaService.tcl module. Its core functionality refers to the web service request / response handling in the service procedure, mainly governed by XML in parsing and XML out generation of the FIXM in and AIXM out payloads. The ZoneCalc.tcl module is responsible for calculating the hazard zone's coordinate verti-ces, derived from the spacecraft's 3d position and orientation, taking into account one-dimensional table interpolation in the hazard area model's database representation, prepared for future vehicle specific tabled values containing aerodynamic coeffi-cients data. C_coding.tcl contains definitions for alternative calculation procedures in embedded C code for the CriTcl C Runtime in Tcl package [4][5], resulting in a more than 30% performance improvement by replacing the CalculateHeading and CalculateHazardZone Tcl procedures by C code equivalents [6].

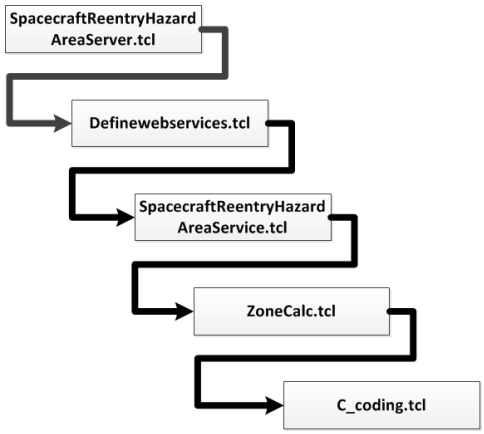

Figure 3. Sourcing structure of the prototype

The principle is shown in Figure 4, taking the CalculateHeading procedure as an example. It checks if CriTcl compiling is sup-ported. If not, the heading calculation is done in pure Tcl with variable Pi and the ToRad procedure, otherwise, the alternative in C code is used with pi and torad defined in C_coding.tcl.

```
# ----------------------------------------------------------------------------
# CalculateHeading --
#     Calculates heading based on two points lat and lon.
#
# Arguments:
#     lat and lon of the two points
#
# Returns:
#     calculated heading
proc CalculateHeading {lat1in lon1in lat2in lon2in} {
   variable Pi
   if {![critcl::compiling]} {
      set LocalOperator1 [expr {cos([ToRad $lat2in]) * sin(([ToRad $lon2in]) - ([ToRad $lon1in]))}]
      set LocalOperator2 [expr {cos([ToRad $lat1in]) * sin([ToRad $lat2in]) - sin([ToRad $lat1in]) \
            * cos([ToRad $lat2in]) * cos(([ToRad $lon2in]) - ([ToRad $lon1in]))}]
      set LocalHeading [expr {atan2($LocalOperator1, $LocalOperator2) * (180 / $Pi)}]
      if {$LocalHeading < 0} {
         set LocalHeadingBuffer $LocalHeading
         set LocalHeading [expr { 360.0 + $LocalHeadingBuffer }]
      }
   } else {

      critcl::cproc c_calcheading {double lat1in double lon1in double lat2in double lon2in} double {
         /* this is C code */
         double localoperator1;
         double localoperator2;
         double localheading;
         localoperator1 = cos(torad(lat2in)) * sin((torad(lon2in)) -  (torad(lon1in)));
         localoperator2 = cos(torad(lat1in)) * sin(torad(lat2in)) - sin(torad(lat1in))
         * cos(torad(lat2in)) * cos((torad(lon2in)) - (torad(lon1in)));
         localheading = atan2(localoperator1, localoperator2) * (180 / pi);
         if (localheading < 0)
         localheading += 360.0;
         return localheading;
      }
      set LocalHeading [c_calcheading $lat1in $lon1in $lat2in $lon2in]
   }
   return $LocalHeading
}
```

Figure 4. CalculateHeading procedure

4. OUTLOOK

Although SESAR SWIM will implement functions that secure information exchanges by infrastructure security, the develop-ment of a SecSWIM security application is planned. It will act as a SWIM participating data analysis sub-system, able to per-form attack identification by SOAP message classification tak-ing into account SOAP characteristics like service identifier, message length, header length, message transaction duration, etc. . The solution is anticipated to benefit from a module based serial-in-the-large-iterative-in-the-small machine learning big data river approach (Figure 5), incorporating a data river→data lake module sequencing (Figure 6).

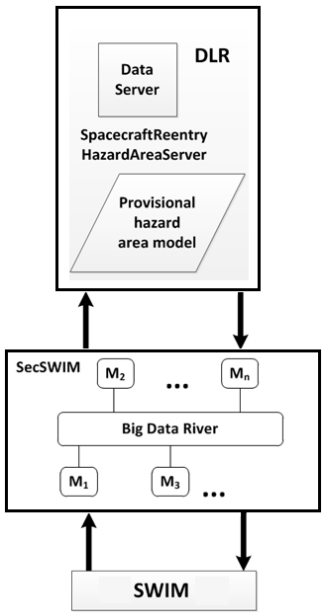

Figure 5. SecSWIM module

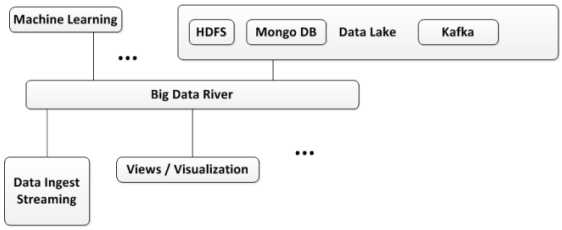

Figure 6. Data River / Data Lake sequence

SecSWIM is foreseen to be developed in Tcl, too. Benefit potential referring the usage of the kafkatcl package [7] and the TensorFlow C++ API (being used inside CriTcl) [8] is under examination.

5. REFERENCES

[1] Seker, R. Moallemi, M., Yapp, J., Towhidnejad, M., and Klein, R. (2014) "Performance Issues in Aircraft Access to the National Airspace SWIM Program" Integrated Communications Navigation and Surveillance Conference, Herndon, VA.

[2] Morlang, F., Hampe, J., Kaltenhaeuser, S., Jakobi, J., Schmitt, D.-R.. (2014). Commercial Space Transportation and Air Traffic Insertion - SESAR Requirements and the European Perspective. In Proc. 1st Annual Space Traffic Management Conference, Daytona Beach FL, USA.

[3] Robledo, L. F. (2004). Analysis and integration of a debris model in the VIRTUAL RANGE PROJECT. A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Modeling and Simulation. Orlando, Florida: College of Engineering and Computer Science at the University of Central Florida.

[4] Landers, S. & Wippler, J-C. (2002). CriTcl - Beyond Stubs and Compilers. In Proc. 9th. Annual Tcl/Tk Conference, Tcl Community Association, Vancouver, Canada.

[5] Kupries, A. (2016). C Runtime In Tcl. In Proc.23rd. Annual Tcl/Tk Conference, Tcl Community Association, Texas, USA.

[6] Morlang, F.. (2017). News from SWIM in Space. In Proc. 9th International Association for the Advancement of Space Safety Conference, Toulouse, France.

[7] KafkaTcl, a Tcl interface to the Apache Kafka distributed messaging system. [Online]. Available: https://github.com/flightaware/kafkatcl. [Accessed 07th Aug. 2018].

[8] TensorFlow C++ Reference. [Online]. Available: https://www.tensorflow.org/api_docs/cc/. [Accessed 07th Aug. 2018].