ATHLA – A TOOL COMMAND LANGUAGE BINDING FOR THE HIGH-LEVEL ARCHITECTURE

Frank Morlang
Institute of Flight Guidance
ATHLA – A Tool command language binding for the High-Level Architecture

Overview

• Introduction
• Motivation
• Approach
• Use Case
• Outlook
ATHLA – A Tool command language binding for the High-Level Architecture

Introduction

• HLA (High Level Architecture)
  • IEEE standard for distributed simulation
  • Purpose → Interoperability and Reuse
  • Started in the 1990s → U.S. DoD
  • Within NATO → STANAG 4603
  • Actual version: HLA 1516-2010
ATHLA – A Tool command language binding for the High-Level Architecture

Introduction

• HLA (High Level Architecture)
  • Architecture
    • Run-time Infrastructure (RTI)
  • Federates
  • Federation Object Model (FOM)
ATHLA – A Tool command language binding for the High-Level Architecture

Introduction

• HLA (High Level Architecture)
  • IEEE Std 1516-2010 Framework and Rules
    • 10 architectural rules
  • IEEE Std 1516.1-2010 Federate Interface Specification
    • specifies the services (provided C++, Java APIs and Web Services)
  • IEEE Std 1516.2-2010 Object Model Template Specification
    • Format of the FOM
Motivation

• Successful Tcl realizations in the past
  • Interfacer to X-Plane flight simulator
  • Apache Kafka consumer and producer
  • RabbitMQ sender and receiver
  • ActiveMQ sender and receiver
  • Different simulation components
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

HLA binding to Tcl built on top of a commercial Java API by usage of the JBlend package.
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

![Diagram showing hlaTcl, TclOO, JBlend, TclHLA_pitch.jar, Source Packages, tclhla, and TclFederate.java]
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Important libraries to use in TclHLA_pitch:

- **Provided libraries of the commercial RTI (Pitch)**
- **tclJBlend.jar of the JBlend Tcl package**
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

The „root“:
• in TclFederate.java

```java
public class TclFederate extends NullFederateAmbassador {
    private RTIambassador _rtiAmbassador;
    private Interp tclInterp;
    private EncoderFactory _encoderFactory;
    ...
    ..
    .
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

The „root“:

• in package hlaTcl.tcl

  oo::class create ::hlaTcl::Hla {
    variable Federate
    constructor {x y} {
      variable JarLocation [file dirname [file normalize [info script]]]
      package require Jblend
      append JarLocation /pitch
      set ::env(TCL_CLASSPATH) $JarLocation
      java::import tclhla.TclFederate
    }
  }
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Initialization:
- in TclFederate.java

```java
public void ini(String rtihost, String federationname, Interp interp) {
    tclInterp = interp;
    String settingsDesignator;
    settingsDesignator = "crcAddress=" + rtihost;
    _rtiAmbassador.createFederationExecution(federationname, urlArray, "HLAfloat64Time");
    ...
    ..
    .
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Initialization:
  • in package hlaTcl.tcl

```tcl
method createFederationExecution {targetaddress federationname} {
    set Federate [java::new TclFederate]
    $Federate ini $targetaddress $federationname [java::getinterp]
    ...
    ..
    .
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Function example:
• in TclFederate.java

```java
public void enable_TimeConstrained() {
    try {
        _rtiAmbassador.enableTimeConstrained();
    } catch (InTimeAdvancingState | RequestForTimeConstrainedPending |
             TimeConstrainedAlreadyEnabled | SaveInProgress |
             RestoreInProgress | FederateNotExecutionMember |
             NotConnected | RTIinternalError ex) {
        Logger.getLogger(TclFederate.class.getName()).log(Level.SEVERE,
                      null, ex);
    }
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Initialization:
- in package hlaTcl.tcl

```
method enableTimeConstrained {} {
    $Federate enable_TimeConstrained
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Callback example:
- in TclFederate.java

```java
@Override
public final void objectInstanceNameReservationSucceeded(String objectName) {
    synchronized (_reservationSemaphore) {
        _reservationComplete = true;
        _reservationSucceeded = true;
        _reservationSemaphore.notifyAll();
        successfullyReservedObject = objectName;
        Notifier n = tclInterp.getNotifier();
        TclEvent t = new EvalObjectInstanceNameReservationSucceededEvent();
        n.queueEvent(t, TCL.QUEUE_TAIL);
        t.sync();
    }
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

**Approach**

Callback example:

- in TclFederate.java

```java
class EvalObjectInstanceNameReservationSucceededEvent extends TclEvent {
    @Override
    public int processEvent (int flags) {
        try {
            tclInterp.eval(tclObjectInstanceNameReservationSucceededScript);
        }
        catch (TclException x) {}  
        return 1;
    }
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Callback example:
- in TclFederate.java

```java
public void set_objectInstanceNameReservationSucceededScript(String objectinstancenamereservationsucceededscript) {
    tclObjectInstanceNameReservationSucceededScript = objectinstancenamereservationsucceededscript;
}
```
Approach

Callback example:

- in package hlaTcl.tcl

```tcl
method setObjectInstanceNameReservationSucceededScript {thescript} {
    $Federate set_objectInstanceNameReservationSucceededScript
    $thescript
}
```
ATHLA – A Tool command language binding for the High-Level Architecture

Approach

Callback example:
• in tcl file using package hlaTcl.tcl

```tcl
proc objectInstanceNameReservationSucceededCallbackReceive {} {
    global reservationComplete
    set reservationComplete 1
    puts "CALLBACK: Object instance name reservation succeeded!"
    return
}

$obj setobjectInstanceNameReservationSucceededScript \
    objectInstanceNameReservationSucceededCallbackReceive
```
ATHLA – A Tool command language binding for the High-Level Architecture

Use Case
ATHLA – A Tool command language binding for the High-Level Architecture

Outlook