NASA Lessons Learned using

High Level Architecture (HLA)

in

Engineering Design Analysis

David Teare
NASA Marshall Space Flight Center
David.A.Teare@nasa.gov
256-544-2075

- **Agenda**
  - **Marshall Space Flight Center's use of HLA in Engineering Design Analysis**
  - **An Experienced Based HLA Interface**
  - **Example Engineering Design Analysis Federation**
  - **Example Federation HLA Performance**
  - **Framework to support highly coupled federates**
  - **Federate Integration “best practices”**
  - **Questions**

- **NASA MSFC use of HLA in Engineering Design Analysis**

  - **Marshall uses HLA to run time based simulations representing the functional performance of a launch vehicle. The HLA federation is the vehicle subsystems and environment models**

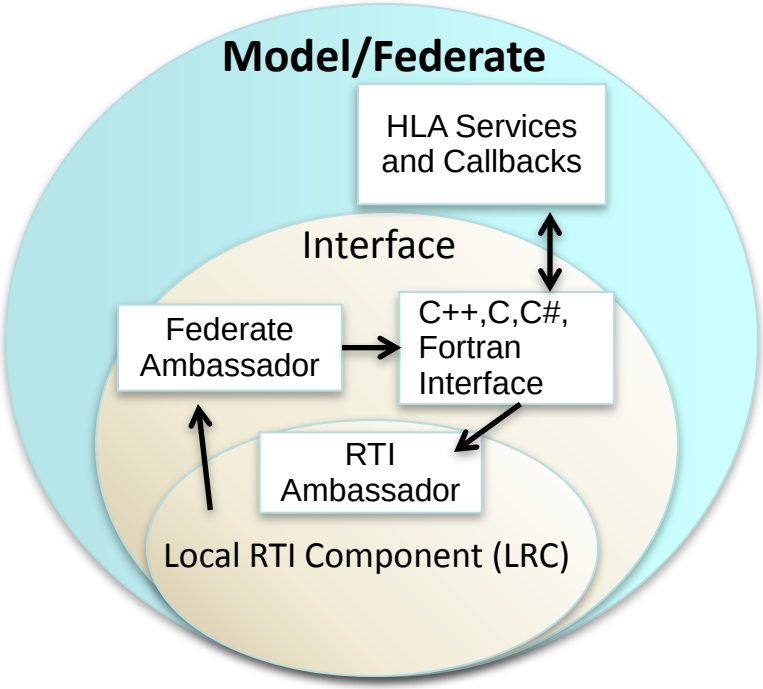
  - **The main purpose of this federation is to provide an integrated vehicle and environment to test the Guidance, Navigation, and Control System logic and Mission and Fault Management logic that is will reside on the launch vehicle's onboard computer**

  - **The launch vehicle's subsystem models represent the current understanding of the subsystem design (e.g. mass, staging, avionics, propulsion). The environment models represent the vehicle's operational environment (earth, atmosphere, winds, etc). The subsystem models are developed by the subsystem designers and the models can be in languages such as C, C++, Fortran, C# and run on Windows, OS X, Linux. We do not modify the model code in order to leverage the verification and validation of the code**

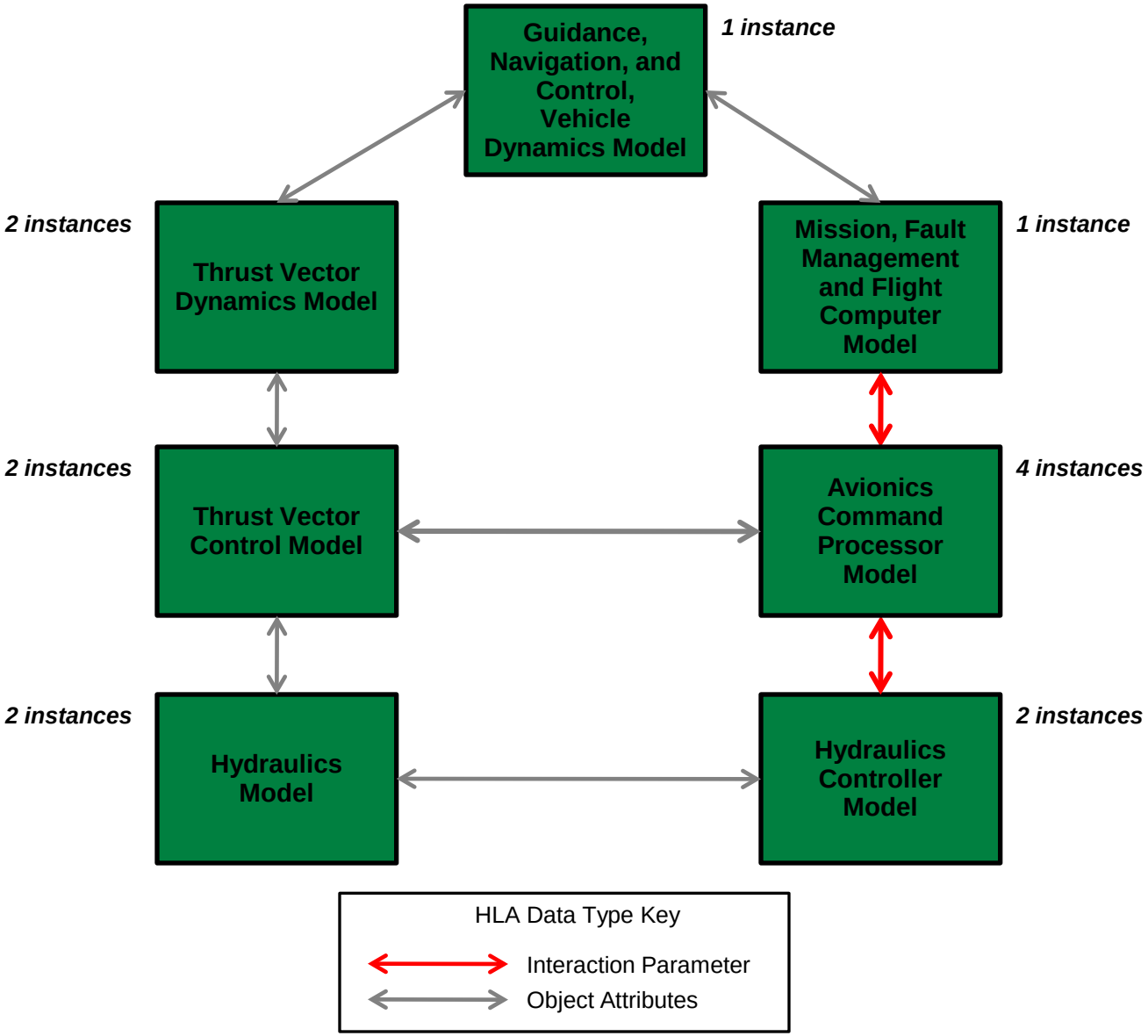
- **NASA MSFC use of HLA in Engineering Design Analysis**
  - **The RTI we are using is from the open source PORTICO project, website is http://www.porticoproject.org/**
  - **Java based, HLA 1.3ng compliant**
  - **PORTICO comes with a Java and C++ Application Programming Interface (API)**
  - **Hosted on an internal NASA TCP/IP network using tunneling since message encryption required**
  - **For single host execution, we are still investigating the use of TCP without accessing network hardware (just using the loopback device)**

- **NASA MSFC use of HLA in Engineering Design Analysis**
  - **The HLA specification contains more than 200 functions and supporting data types to cover a broad range of federation time and data management requirements. Experience has shown that many of these functions are not used. In fact, only a subset are used as a sequence of calls and callbacks**
  - **The HLA specification is a C++ specification and includes C++ exceptions and classes that aren't compatible with many NASA legacy simulations. A model written in C or FORTRAN cannot interface directly with the RTI Ambassador, nor provide a Federate Ambassador class**
  - **The HLA specification forbids concurrent access, meaning that federates cannot call an RTI function during a Federate Ambassador callback. Chains of events that can cause this to happen are often hard to foresee and require caching and queuing**

- **Experience Based HLA Interface**
  - **The Interface implements the 'low-level' RTIAmbassador-to-FedAmbassador interface, and provides a simplified API (in C++, C, C#, and Fortran) to federates**
  - **Two-way communication is implemented as 'callbacks', using function pointers which are supported in the C, C++, C#, and FORTRAN languages**
  - **Setting up HLA and setting up time management are two single function calls as opposed to several calls between the RTIAmbassador and the FederateAmbassador**
  - **The interface is 23 functions and 4 types. Function arguments are simple types, arrays, and pointers, and return 0 upon success**
  - **The Interface queues time stamp ordered messages that arrive during time advancements and delivers them to federates after the advancement is complete to prevent concurrent access**



Example Engineering Design Analysis Federation (14 Federates)



- **Example Federation HLA Performance**

  - **For the example, we have had up to 14 federates running under time management with each federate both constrained and regulating.**
  - **These federates are running with time steps of 0.0001 sec (10,000 Hz) for 140 seconds of run time. One run take more than 3 hours of clock time to complete.**
  - **Distributing the federates over multiple hosts does not improve performance because of the highly coupled federates sending and receiving data at the small time interval**
  - **Faster inter process communication (IPC) is needed to reduce the Design Analysis run time for the highly coupled federates**

- **Framework to support highly coupled federates**

  - **We have developed a initial version of a shared memory framework (SHM_FW) to address the IPC issue we are experiencing**
  - **This framework allows the models/federates to publish/subscribe using a common memory location. The read/write of memory is extremely fast**
  - **The SHM_FW will use the exact same method/function calls that our HLA Interface uses so that we can switch frameworks by recompiling the code using pre-compile flags. No code changes are required**
  - **However, the Federation will be limited to running on a single machine**
  - **Preliminary results have shown a factor of 10 performance improvement**

- **Federate Integration process “best practices”**

–If you are working with a model you did not develop (delivered model) it is very important to have:

- **A point of contact for the model**
- **Test cases (Makefile, build scripts, input/output history) delivered with the model**
- **Documented Model interface requirements**
- **Any model verification and validation information/data**

–If you developed the model:

- **Define and document model test cases**
- **Document the required interface**
- **Verify and validate your model and document the results**

–If possible, have an independent reference or benchmark to compare with your integrated federation results. At a minimum have an understanding what should and should not happen

- **Questions**