JSC XXXXX-DD.mm.20

Trick High Level Architecture
TrickHLA
Product Requirements

Simulation and Graphics Branch (ER7)
Software, Robotics and Simulation Division
Engineering Directorate

Package Release TrickHLA v3.0.0 - Beta

Document Revision 1.0
June 2020

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC XXXXX-DD.mm.20

Trick High Level Architecture
TrickHLA
Product Requirements

Document Revision 1.0
June 2020

Edwin Z. Crues
and
Daniel E. Dexter

Simulation and Graphics Branch (ER7)
Software, Robotics and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

TrickHLA is a middleware model package that provides an interface framework for enabling IEEE-1516 High Level Architecture (HLA) capabilities in simulations developed in the Trick Simulation Environment. TrickHLA allows a developer to concentrate on simulation development without needing to be an HLA expert. The TrickHLA model is data driven and provides a simplified API making it relatively easy to take an existing Trick-based simulation and make it HLA capable.

Contents

Chapter 1

Introduction

The objective of TrickHLA is to simplify the process of providing simulations built with the Trick Simulation Environment[6] with the ability to participate in distributed executions using the High Level Architecture (HLA)[10]. This allows a simulation developer to concentrate on the simulation and not have to be an HLA expert. TrickHLA is data driven and provides a simple API making it relatively easy to take an existing Trick simulation and make it HLA capable.

1.1 Identification of Document

This document describes the TrickHLA model developed for use in the Trick Simulation Environment. This document adheres to the documentation standards defined in NASA Software Engineering Requirements Standard [5].

1.2 Scope of Document

This document provides information on the requirements for TrickHLA.

1.3 Purpose and Objectives of Document

The purpose of this document is to define the set of requirements that the TrickHLA must achieve to be compatible with Federate Inferface Specification of the IEEE Standard for Modeling and Smulation (M&S) High Level Architecture (HLA) [9].

1.4 Documentation Status and Schedule

The information in this document is current with the TrickHLA v3.0.0 - Beta implementation of the TrickHLA. Updates will be kept current with module changes.

Author	Date	Description
Edwin Z. Crues	June 2020	TrickHLA Version 3

Revised by	Date	Description

1.5 Document Organization

This document is organized into the following sections:

Chapter 1: Introduction - Identifies this document, defines the scope and purpose, present status, and provides a description of each major section.

Chapter 2: Related Documentation - Lists the related documentation that is applicable to this project.

Chapter 3: Requirements - Presents the requirements for the TrickHLA.

Bibliography - Informational references associated with this document.

Chapter 2

Related Documentation

2.1 Parent Documents

The following documents are parent to this document:

- *Trick High Level Architecture (TrickHLA)* [2]

2.2 Applicable Documents

The following top level documents are applicable to this document:

- *TrickHLA Product Specification* [3]
- *TrickHLA User Guide* [4]
- *TrickHLA Inspection, Verification, and Validation* [1]

The following specific documents are applicable to this document:

- *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Inferface Specification* [9]
- *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification* [11]

The following additional documents are applicable to this document:

- *Trick Simulation Environment: Installation Guide* [7]
- *Trick Simulation Environment: Tutorial* [8]
- *Trick Simulation Environment: Documentation* [6]
- *NASA Software Engineering Requirements* [5]

Chapter 3

Requirements

3.1 General Requirements

This section identifies general requirements for the TrickHLA.

Requirement TrickHLA_1: Documentation

Requirement:
The documentation for the model shall include

1.1 Software requirements specification.

1.2 Software, interface, and software version descriptions.

1.3 Software test procedures and results.

1.4 User Guide.

Rationale:
The listed items are needed to comply with NASA NPR 7150.2 as a Class C product.

Verification:
Inspection

Requirement TrickHLA_2: Header File Trick Header

Requirement:
All header files associated with the model shall have an appropriate Trick header. The Trick header for a header file shall include

2.1 Purpose A brief description of the file.

2.2 References A list of applicable references that describe the model.

2.3 Assumptions and limitations A list of the assumptions made in developing the model and any limitations on the use of the model.

2.4 Programmer A list of the developers who created or modified the file.

Rationale:

- The presence of a Trick header in a header file indicates that Trick should process the file.
- Properly documenting the TrickHLA package models is a key goal of the TrickHLA verification, validation, and documentation task.
- Maintaining a version history is good programming technique and is mandatory per NPR 7150.2.

Verification:

Inspection

Requirement TrickHLA_3: Trick Comments for Enumerated Types

Requirement:

Each tag defined in a enumeration type in a model header file *should* have a comment describing the tag that follows the tag declaration.

Rationale:

Short tag names may not suffice in establishing the meaning of the tag.

Verification:

Inspection

Enumerated types that fail to meet this optional requirement shall be noted as such in the model verification document.

Requirement TrickHLA_4: Trick Comments for Data Structures

Requirement:

Each element of a data structure defined in a model header file shall have a Trick-compliant comment describing the element that follows the element declaration.

Rationale:

The element comment is required by Trick.

Verification:

Inspection

Requirement TrickHLA_5: Source File Trick Headers

Requirement:

Each externally visible function defined in the source files associated with the model shall have an appropriate Trick header. The Trick header for a function shall include

5.1 Purpose A brief description of the function.

5.2 References A list of applicable references that describe the function.

5.3 Assumptions and limitations A list of the assumptions made in developing the function and any limitations on the use of the function.

5.4 Class The default Trick job classification of the function.

5.5 Library dependency A list of the object files upon which the function depends, starting with the current file.

5.6 Programmer A list of the developers who created or modified the file.

Rationale:

- The Trick header that precedes a function indicates that the function is available for use in a simulation *S_define* file.
- Properly documenting the TrickHLA package models is a key goal of the TrickHLA verification, validation, and documentation task.
- Maintaining a version history is good programming technique and is mandatory per NPR 7150.2.

Verification:

Inspection

Requirement TrickHLA_6: Trick Comments for Function Definitions

Requirement:

Each function shall be commented with a Trick-compliant set of comments that describe the return value from the function and that describe the nature of the arguments passed to the function.

Rationale:

The function definition comments are required by Trick.

Verification:

Inspection

Requirement TrickHLA_7: HLA Federate Interface

Requirement:
The TrickHLA model shall use the HLA federate interface specification defined by IEEE standard 1516.1-2010 [9].

Rationale:
The TrickHLA model is being designed to use the HLA interfaces specified in the IEEE 1516.1-2010 standard [9].

Verification:
Inspection

3.2 Data Requirements

This section identifies requirements on the data represented by the TrickHLA. These as-built requirements are based on the TrickHLA data definition header files.

Requirement TrickHLA_8: Primitive Data Types

Requirement:
The TrickHLA model shall support the C and C++ primitive data types supported by the Trick simulation environment.

8.1 bool
The *bool* Boolean data type for C++ shall be supported.

8.2 char
The *char* integer data type for both C and C++ shall be supported.

8.3 unsigned char
The *unsigned char* integer data type for both C and C++ shall be supported.

8.4 short
The *short* integer data type for both C and C++ shall be supported.

8.5 unsigned short
The *unsigned short* integer data type for both C and C++ shall be supported.

8.6 int
The *int* integer data type for both C and C++ shall be supported.

8.7 unsigned int
The *unsigned int* integer data type for both C and C++ shall be supported.

8.8 long
The *long* integer data type for both C and C++ shall be supported.

8.9 unsigned long
The *unsigned long* integer data type for both C and C++ shall be supported.

8.10 long long
The *long long* integer data type for both C and C++ shall be supported.

8.11 unsigned long long
The *unsigned long long* integer data type for both C and C++ shall be supported.

8.12 float
The *float* floating-point data type for both C and C++ shall be supported.

8.13 double
The *double* floating-point data type for both C and C++ shall be supported.

Rationale:
The C and C++ primitive data types are expected to be needed by the various simulation developers.

Verification:
Inspection or Test

Requirement TrickHLA_9: Static Arrays of Primitive Data Types

Requirement:
The TrickHLA model shall support static arrays of C and C++ primitive data types.

Rationale:
Static arrays of C and C++ primitive data types are expected to be needed by the various simulation developers.

Verification:
Inspection or Test

3.3 Functional Requirements

This section identifies requirements on the functional capabilities provided by the TrickHLA. These as-built requirements are based on the TrickHLA source files.

Requirement TrickHLA_10: Data Driven

Requirement:
The TrickHLA model shall be data driven so that it can be configured in the Trick Simulation Input file.

Rationale:

A data driven design will allow the TrickHLA model configuration to be modified without needing to recompile the simulation.

Verification:

Inspection

Requirement TrickHLA_11: HLA Big and Little Endian

Requirement:

The TrickHLA model shall be able to automatically translate primitive data types to either Big or Little Endian based on the specified Endianess of the corresponding HLA attribute or parameter.

Rationale:

The Federation Object Model (FOM) used by HLA defines what data, type, Endianess, and encoding to be used to exchanged data between federates. The TrickHLA model must be able to translate to and from the data types specified in the FOM used for exchanging data using HLA.

Verification:

Inspection or Test

Requirement TrickHLA_12: HLA Encoding

Requirement:

The TrickHLA model shall support the HLAunicodeString, HLAASCIIstring, and HLAopaqueData encodings as specified in the HLA Object Model Template Specification [11].

Rationale:

Strings may be encoded at the HLA interface using either HLAunicodeString, HLAASCIIstring, or HLAopaqueData. The TrickHLA model uses the HLAunicodeString and HLAASCIIstring for encoding strings that need to pass through the HLA interface. The TrickHLA model uses the HLAopaqueData encoding for encoding a generic array of bytes.

Verification:

Inspection or Test

Requirement TrickHLA_13: Time Advancement

Requirement:

The TrickHLA model shall automatically handle time advancement.

Rationale:

The HLA standard provides an interface for managing time advancement among the federates in the federation and should be coordinated with the simulation.

Verification:
Inspection or Test

Requirement TrickHLA_14: Lag Compensation

Requirement:
The TrickHLA model shall provide an interface to the simulation to allow for none, send-side, or receive-side lag compensation.

Rationale:
Lag compensation is a technique where the latency between the time data is sent a one federate and received at another can be compensated for by the simulation builder.

Verification:
Inspection or Test

Requirement TrickHLA_15: Interactions

Requirement:
The TrickHLA model shall provide an interface to the simulation to allow for sending and receiving of interactions as either Receive Order or Timestamp Order.

Rationale:
Interactions can be either an asynchronous or synchronous communication of data between federates. Interactions are expected to be needed by the various simulation developers.

Verification:
Inspection or Test

Requirement TrickHLA_16: Ownership Transfer

Requirement:
The TrickHLA model shall provide an interface for transfering ownership of data attributes between federates.

Rationale:
Ownership transfer of data attributes is expected to be needed by the various simulation developers.

Verification:
Inspection or Test

Requirement TrickHLA_17: Dynamic Initialization

Requirement:
The TrickHLA model shall provide an interface for dynamically initializing the data of a simulation.

Rationale:

It is expected that the federates will need to exchange initialization data before the simulation starts.

Verification:

Inspection or Test

Requirement TrickHLA_18: Automatic Simulation Startup

Requirement:

The TrickHLA model shall automatically synchronize the startup of a distributed simulation given a list of federates required to run the simulation.

Rationale:

Ensuring that all the required federates have joined the federation and start executing the simulation at the same time is expected to be needed by the various simulation developers.

Verification:

Inspection or Test

Requirement TrickHLA_19: Pack / Unpack of Simulation Data

Requirement:

The TrickHLA model shall provide an interface to allow preprocessing (pack) of data before it is sent through HLA interface and post-processing (unpack) the data received from the HLA interface.

Rationale:

A simulation developer may need to perform calculations or other work before data is sent to and/or received from the HLA interface.

Verification:

Inspection or Test

Requirement TrickHLA_20: Object Deleted Notification

Requirement:

The TrickHLA model shall provide an interface to allow notification of an object being deleted from the federation.

Rationale:

A simulation developer may need to carry out specific actions when a data object is deleted from the federation.

Verification:

Inspection or Test

Requirement TrickHLA_21: Federation Restore

Requirement:
The TrickHLA model shall provide an interface to allow the user to programmatically request a federation restore from their trick model.

Rationale:
A simulation developer may need to request a federation restore from their trick model at any time.
`NOTE`: In the current implementation, the programmatic call to restore a federation cannot happen during federation execution; it can only occur when the federation is starting up.

Verification:
Inspection or Test

Requirement TrickHLA_22: Federation Save

Requirement:
The TrickHLA model shall provide an interface to allow the user to programmatically request a federation save from their trick model.

Rationale:
A simulation developer may need to request a federation save from their trick model at any time.

Verification:
Inspection or Test

Requirement TrickHLA_23: Conditional sending of attributes

Requirement:
The TrickHLA model shall provide an interface to allow the user to programmatically identify when to send attributes from their trick model.

Rationale:
A simulation developer may need send only changed attributes to other federates rather then burning up bandwitdh to send the full complement of attributes when only a few may have actually changed.

Verification:
Inspection or Test

Requirement TrickHLA_24: Multiple verbosity levels

Requirement:
The TrickHLA model shall provide an interface to allow the user to specify multiple levels of verbosity from TrickHLA.

Rationale:

A simulation developer may wish to see different levels of information from the inner workings of TrickHLA software to aid in debugging simulation problems.

Verification:

Inspection or Test

Bibliography

[1] Edwin Z. Crues. *TrickHLA Inspection, Verification, and Validation*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, June 2020.

[2] Edwin Z. Crues. *Trick High Level Architecture (TrickHLA)*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, June 2020.

[3] Edwin Z. Crues. *TrickHLA Product Specification*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, June 2020.

[4] Edwin Z. Crues. *TrickHLA User Guide*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, June 2020.

[5] NASA. *NASA Software Engineering Requirements*. Technical Report NPR-7150.2C, National Aeronautics and Space Administration, NASA Headquarters, Washington, D.C., August 2019.

[6] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7). Trick simulation environment: Documentation. Trick Wiki Site: https://nasa.github.io/trick/documentation/Documentation-Home, May 2020. Accessed 18 May 2020.

[7] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7). Trick simulation environment: Installation guide. Trick Wiki Site: https://nasa.github.io/trick/documentation/install_guide/Install-Guide, May 2020 (accessed May 18, 2020). Accessed 18 May 2020.

[8] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7). Trick simulation environment: Tutorial. Trick Wiki Site: https://nasa.github.io/trick/tutorial/Tutorial, May 2020 (accessed May 18, 2020). Accessed 18 May 2020.

[9] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Inferface Specification*. Technical Report IEEE-1516.1-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.

[10] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.* Technical Report IEEE-1516-2010, The Institute of Electrical and Electronics Engineers, 2 Park Avenue, New York, NY 10016-5997, August 2010.

[11] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification.* Technical Report IEEE-1516.2-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.