Progress Toward a Format Standard for Flight Dynamics Models

E. Bruce Jackson
NASA Langley Research Center
Dynamic Systems and Control Branch, MS308
Hampton, Virginia 23681
757-864-4060
e.b.jackson@nasa.gov

Bruce L. Hildreth
SAIC
22299 Exploration Drive, Suite 200
Lexington Park, Maryland 20653
301-863-5077
bruce.hildreth@saic.com

ABSTRACT: *In the beginning, there was FORTRAN, and it was... not so good. But it was universal, and all flight simulator equations of motion were coded with it. Then came ACSL, C, Ada, C++, C#, Java, FORTRAN-90, Matlab/Simulink, and a number of other programming languages. Since the halcyon punch card days of 1968, models of aircraft flight dynamics have proliferated in training devices, desktop engineering and development computers, and control design textbooks. With the rise of industry teaming and increased reliance on simulation for procurement decisions, aircraft and missile simulation models are created, updated, and exchanged with increasing frequency. However, there is no real lingua franca to facilitate the exchange of models from one simulation user to another. The current state-of-the-art is such that several staff-months if not staff-years are required to 'rehost' each release of a flight dynamics model from one simulation environment to another one. If a standard data package or exchange format were to be universally adopted, the cost and time of sharing and updating aerodynamics, control laws, mass and inertia, and other flight dynamic components of the equations of motion of an aircraft or spacecraft simulation could be drastically reduced. A 2002 paper estimated over $ 6 million in savings could be realized for one military aircraft type alone. This paper describes the efforts of the American Institute of Aeronautics and Astronautics (AIAA) to develop a standard flight dynamic model exchange standard based on XML and HDF-5 data formats.*

1. Introduction

This paper focuses on work that members of the American Institute of Aeronautics and Astronautics (AIAA) have performed to develop standards on flight simulation models. The AIAA is the largest aerospace engineering professional society [1]. It consists of approximately 31,000 aerospace professionals throughout the United States and the world. As part of the AIAA there are many standing technical committees and this paper report on work performed by the Modeling and Simulation Technical Committee.

Specifically, this standard is focused on dynamic models, those models and functions that make the simulation fly like an airplane. However the work reported here has some application to all models, specifically in the format of function tables and time history data created by simulation or test instrumentation. For the purpose of this paper, a flight dynamics model will be defined as those simulation model components that change when you change a simulation from one aircraft type to another. This is typically the aerodynamics, engine, mass and inertia, and ground reactions. This standard does not presently address controls, electrical and hydraulic systems, and avionics. Standardizing the exchange of those model components is planned for the future.

As any standards organization understands, a standard cannot just be “thrown out” into the community but needs to be socialized, tested and optimized to the user community’s requirements. The purpose of socialization is to explain its applications and use to the community, get feedback from the community on how to make it most effective, coordinate with other standards activities, and help create the user base which will make the standardization worth while. It is to this end that this paper has been generated for the Simulation Interoperability and Standards Organization (SISO). The AIAA has communicated with various personnel associated with SISO to verify that SISO is not working in a parallel path. Hopefully readers of this paper will agree and take this opportunity to learn about and provide feedback on this standard.

Furthermore, in addition to soliciting SISO members for their ideas and feedback, the AIAA would like to solicit SISO endorsement of the standard. SISO in general develops IEEE standards. AIAA is an ANSI standard member. Further investigation will be required to see if a standard developed by one standards organization (ANSI in this case) can be tolerated by another standards organization (IEEE). Hopefully we can all work together to accept and improve the standard, consequently increasing the potential to improve productivity throughout our industry.

1.1 Goal: A Standard for the Exchange of Flight Dynamics Models

The goal of the standard is to dramatically facilitate the process of exchanging a model of an aircraft from one simulation facility to another. The benefit to the aerospace community will be the potential for a significant improvement in collaboration within the simulation community of a specific aircraft type, an F-16 model type, for example.

1.1.1 Motivation for a Flight Dynamics Model Standard

Frankly, there is no presently accepted method for exchanging models between simulation facilities. Virtually every simulation site in the aerospace community has different simulation architectures, data formats, and software. This is true not only for specific models of specific aircraft, but even for models and software that could be invariant across the aerospace community, such as equations of motion, axis transformations, numerical integration techniques, atmosphere models, etc. This lack of standardization is true even within the same government departments and within large corporations. Each one has their own models and methods for performing flight and space simulation.

As a result, every time a model is sent from one facility to another, unique import and export software must be written and tested to facilitate the model exchange. Figure 1.1 below represents what is typical in industry. Every time one simulator site, for example a research, development, test and evaluation (RDT&E) simulator at site A, wants to send a model or even components of a model such as an updated aerodynamic function to site B, they have to write import and export tools specifically for that site. And if they want to send the same information to site C they have to write different import and export tools. This is obviously very inefficient. If there are many different sites (10-30 sites are not at all unusual, in fact often for most large weapon systems there are more then 10 sites of simulators with different architectures [2]) the import/export task becomes prohibitive. Therefore, normally it is simply not done. Each site does its own model configuration control and has its own strengths and weaknesses in the model. The real inefficiency is that when each site works independently, and a specific site learns how to improve the model, the other sites do not benefit, and vice-versa. Additionally, even when the model is sent between sites there is typically very little verification or validation data sent.

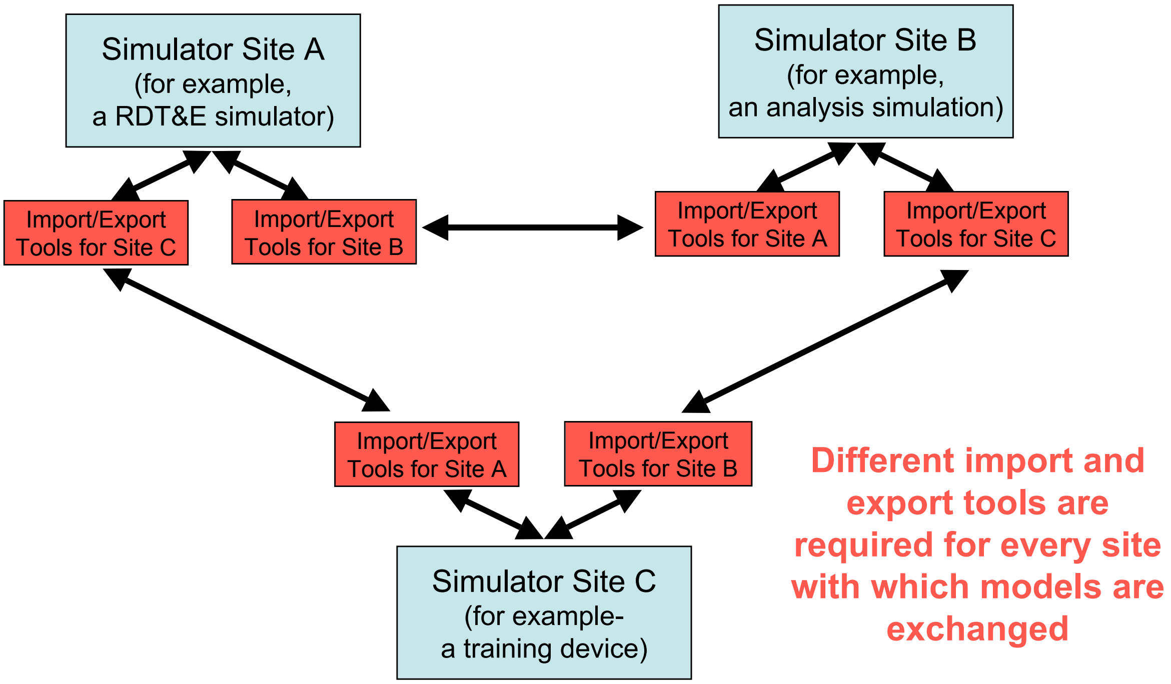

Figure 1.1 The existing problem inhibiting the exchange of models, and therefore collaboration, between sites.

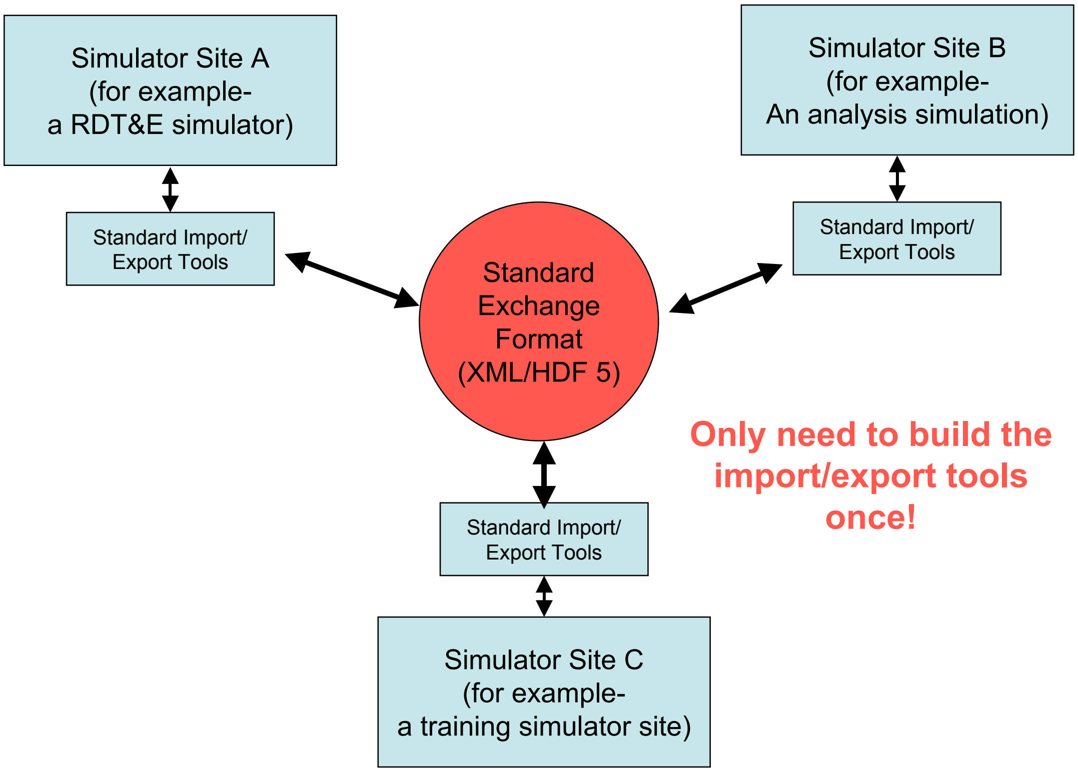

Figure 2.1 The simulation model exchange standard allows one set of tools to be used to exchange models between any other facility-facilitating collaboration between simulation sites

2. One solution - a standard format to exchange models and validation data

The proposed ANSI standard establishes the **definition of the information** and **format** used to exchange air vehicle simulations and validation data between disparate simulation facilities. This standard is not meant to require facilities to change their internal formats or standards. With the concept of an exchange standard, facilities are free to retain their well-known and trusted simulation hardware and software infrastructures. The model is exchanged through the standard, so each facility only needs to create import/export tools to the standard once. Then they can use these tools to exchange models with any facility at minimal effort, instead of making unique import/export tools for every facility that they exchange with. Over time, as tools improve, an organization might logically evolve to using the standard as their internal format. However, they would not be forced to do so on a specific schedule or to qualify for any specific program. This makes the standard very attractive to all organizations. Figure 2.1 illustrates this concept.

The standard includes **definition of the information** in order to clarify the information exchanged. Such clarification includes axis systems referenced, units, and sign conventions used. The **format** (either XML or HDF) defines the mechanism used to facilitate automation of the exchange of the information. Using the definitions in the standard, a **list of simulation variable names and axis system** is included. This list of standard variables names further simplifies the exchange of information, but is not required.

2.1 Present Applications of the Standard

The standard is presently nearing its first release. It has excellent application in the exchange of aerodynamic models. Aerodynamic simulation models typically consist of multi-dimensional function tables and very simple math to look up these tables and compute aerodynamic forces and moments in specific axis systems, usually a body axis system. The standard handles this very well and provides a very flexible method for meeting the user’s needs. The function table standard is well developed and includes capability to include the provenance (where the data came from) all the way down to an individual data point basis. Additionally it allows statistical information to be included with the data to better facilitate Monte Carlo and other uncertainty studies.

Since aerodynamic functions for nonlinear simulations may have many multidimensional functions with literally millions of data points, the management of these aerodynamic models is a significant issue. It is very difficult to keep track of where the data came from for a specific model and the specific aerodynamic functions are frequently changed to better match expected response and validation data. It is this area where the standard will be of immediate benefit to the community. When a specific aerodynamic function is improved to match the response of the aircraft it will be easy to promulgate throughout the rest of the simulation community of that aircraft. Unfortunately this is presently not done on a routine basis.

Weight and balance models are typically simple math and table look-ups and therefore, like aerodynamic models, the standard is well suited in transferring this information.

For engine models the nonlinear function representation is well taken care of by the function table look ups but the math for the model becomes more complicated; this first version of the standard supports arbitrary propulsion calculations but does not support dynamic states or engine control systems.

Other components of the model such as ground reactions, control systems, electrical and hydraulic systems, and avionics models are not addressed by the standard. Any nonlinear functions and simple math routines could be, of course, accommodated by the standard, but the normal higher complexity of the differential equations required in these models will be supported in a later version of the standard.

2.2 Components of the standard

The standard is really a method of communicating flight model information. This work has attempted to draw upon other existing work. For example, the axis system is consistent with published AIAA recommended practices [3] and DIS axis system definitions [4].

The standard consists of four sections and an introduction. The four sections are:

- Standard axis systems
- Standard variable names
- Standard function table formats
- Standard validation data formats (time history or frequency domain data format)

2.2.1 Standard axis systems

The axis system definition is complete and based on AIAA recommended practice [3] and DIS standards [4] already published. The standard is the overlap of both of these standards. The variable names reference the axis systems used.

Axis system standards also are reflected in the variable naming convention. When applicable, the axis system is included in the variable name. The following systems are proposed:

- Geocentric Earth Centered/ Earth Fixed (ECEF) Axis System
- Body Axis Coordinate System
- Flat Earth Axis System

2.2.2 Standard variable names

It is important to understand that this is an interchange standard. Part of the standard is variable definitions and axis system definitions to facilitate the interchange of information. These components of the standard are not to require people to use certain variable names or axis systems but make it easy to communicate information via standard names and standard axis systems when desired.

It is much simpler to exchange information when the exchange is through common definitions of variables, units, sign convention, and axis systems. Without the standard variable names and axis systems, not only does the data itself have to be exchanged but a complete set of definitions defining what the data is must be exchanged. If the standard variable names and axis systems are used then only the data needs to be exchanged with a tag saying what the variable name is. The standard contains the rest of the information.

The variable naming convention includes a methodology for naming new variables to allow consistency when adding variables to the standard. It is also consistent with object oriented design techniques and structured software.

The variable name standard actually has two distinct parts:

- A list of standard variable names for variables typical of aircraft simulations
- A methodology and convention for defining variables specific to a simulation domain so the list may be expanded at the local and national organizational levels.

The general rules for naming variables and are similar to what is generally used in C and C++ programming:

- Variables shall have meaningful names. Mnemonics will not be used. Standard abbreviations are allowed.
- Distinct words in variable names shall be separated by capital letters.

- Variable names shall not exceed 60 characters in length. Brief but concise names are most effective.
- A short variable name (8 characters) is allowed as an alias.
- The first letter of the variable name is lower case. Units should be all lower case. This reduces ambiguity.

Methodology for Creating New Names

The method for creating a new variable name is as follows.

Each name has up to six components. All components are not required to be used because in many cases they do not apply. These components are:

> *(prefix)_(variable source domain)(axis or reference system)(specific axis or reference)(core name)_(units)*

The purpose of the prefix is to identify the variable in two ways. The prefix is used to denote the key variables in the model, which are the model states and state derivatives. Mathematically, all outputs of the model are derived from these.

It is important to emphasize identification of the states and the inputs are a key factor in simulator software development and maintenance of existing simulation software. This convention should hold true for control states, landing gear states, and any dynamic system in the vehicle.

Examples of standard variables:

`thrustBodyXForce_N`

- `thrust` - the domain of the variable (as opposed to aero for example)
- `body` - indicate the axis system
- `X` - indicates the specific axis
- `Force` - is the core variable name
- `N` - indicates unit of measure (Newtons in this case)

`s_bodyXVelocity_fs-1`

- `s_` - prefix indicates that this is a state variable
- `body` - indicate the axis system
- `X` - indicates the specific axis
- `Velocity` - is the core variable name
- `fs-1` - indicates unit of measure, feet per second.

`bodyRollRate_ds-1`

- The lack of the `s_` prefix indicates that this is not the roll rate state variable
- `body` - indicate the axis system
- `Roll` - indicates the specific axis
- `Rate` - core variable name
- `ds-1` - unit of measure, degrees per second.

2.2.3 Standard function tables

The first and foremost design objective of the Standard Data Table Format was to make a data format that would **include all the information about real multi-dimensional data, not just the data values**. This objective reflects the fact that, in the general case of the independent variables for a multi-dimensional table, the independent variables have different number of breakpoints, different breakpoints, and different valid ranges. A second design objective was to allow the table to contain information on where the data points come from (provenance, via reference), and a confidence interval for the data. Confidence Intervals can be used for Monte Carlo simulations and to mathematically combine two different estimates of the same parameter at the same point. Therefore, confidence statistics are extremely valuable when attempting to update a data set (however the user must be careful as not all confidence intervals are equivalent, or even meaningful). Finally, the table has to be easy to read by the computer and the human being, and be self-documenting as much as possible.

Figure 2.2 presents a fairly standard three-dimensional set of data as is typical of aerodynamic data from flight test or from a wind tunnel. In the example given, lift coefficient is a function of angle-of-attack, Mach number, and a control position. As a more general statement, the function output, dependent variable CLALFA, is dependent on three inputs (the independent variables) alfa, Mach, and delta_s. Close examination of the data given will reveal the following characteristics:

The number of breakpoints of the independent variables varies for each independent variable. Not only are there a different number of angle-of-attack (alfa) breakpoints, but also a different number of Mach number (Mach) and control position (delta_s) breakpoints. The standard defines this as an “ungridded table.”

The values (breakpoints) of the independent variables are different. Again, an “ungridded table.”

The valid ranges of the independent variables are different. (“ungridded table”)

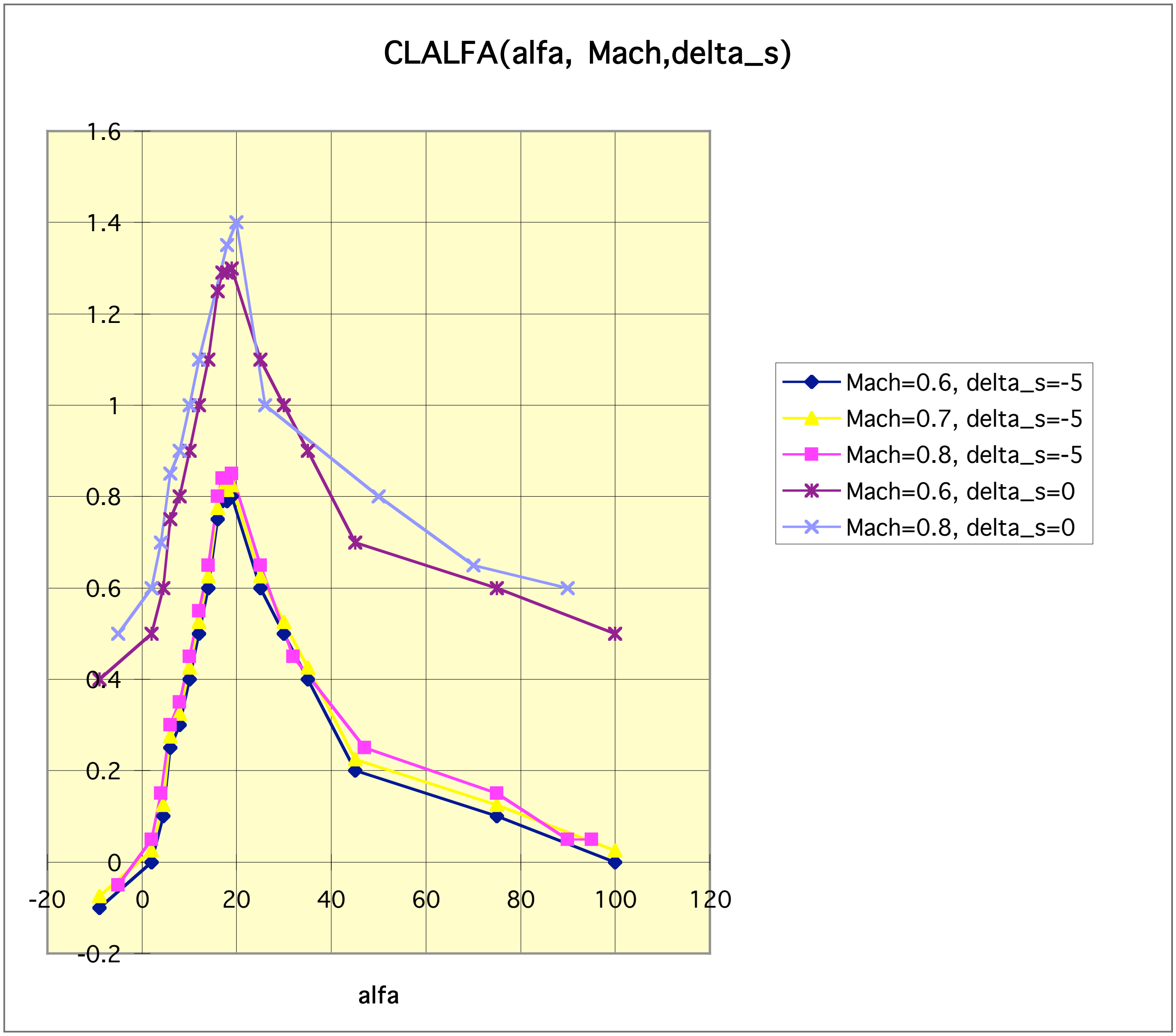

Figure 2.2 An Illustration of a Three-Dimensional Function Table, CLALFA (alfa, Mach, delta_s)

The above three differences are not consistent for all the data. For example, in the example table the alfa breakpoints for Mach = 0.6 and Mach = 0.7 for delta S = –5 are identical.

For the function data there is other information not shown in figure 2.2 that is of significant importance to the user, without which the data is not very useful. In general this information is:

- where did the data come from? (For example what report?)
- how is it defined? (For example, is this at a specific altitude? What configuration is it for?)
- what are the engineering units of the output (the dependent variable) and the independent variables?
- what is the sign convention of the independent and dependent variables? (For example, is the control position positive trailing edge up or trailing edge down? Exactly which control surface is it?)
- who created the table? (Not where the data came from, but what person decided that this was the correct data for this table?)
- how has it been modified and for what reason?
- how accurate is the data estimated to be? (Or, mathematically what is the confidence interval of the data?)

- By what method is the data intended to be interpolated? (For example, linear interpolation or bi-cubic spline interpolation?)
- By what method is the data intended to be extrapolated for data with different ranges?

The standard exchange format has data elements that contain all of the above information. It has been implemented in XML as seven Major Elements and is introduced below and discussed in detail in the simulation exchange markup language reference manual [5].

3. Overview of the XML model standard

3.1 XML itself

The eXtensible Markup Language, XML, is an World-Wide Web Consortium (W3C) standard means of encoding information in a human- and machine-readable way, similar to the way information is encoded in hypertext markup language (HTML) for web browsers. XML is extensible in that a set of markup tags and allowable information hierarchy can be designed for each type of data to be encoded; by agreeing to such a grammar (or XML application), data can be freely exchanged between parties.

Several benefits of using XML are immediately available:

- The data document can be validated against the grammar definition to ensure proper encoding; this helps catch encoding mistakes.
- The information in the document can be read (and edited) by humans while remaining usable for machine processing.
- The data can be converted from one XML format to another by use of another XML grammar, known as eXtensible Stylesheet Language, documents. This allows the data to be converted for viewing by any standards-compliant web browser, potentially on-the-fly.
- An increasing number of low-cost or freely available, multi-platform XML editing tools are becoming available; this should ease the adoption of XML for a variety of data encoding applications (including flight dynamic model exchanges).

3.2 The schema (DTD)

A schema, or XML grammar definition, has been developed to handle the static parts of a flight dynamics model [5]. (Work on a schema to handle arbitrary dynamics is pending.) While this schema addresses only part of the problem space, it does cover a large portion thereof, namely the aerodynamic model, which consists of tabular data and buildup equations. The schema also allows incorporation, into the same XML file, of verification data to ensure proper implementation of the model by the recipient.

We have selected the older Document Type Definition (DTD) method of expressing this schema, due to the wider applicability of this format, rather than the newer XML Schema Definition (XSD) language.

The schema provides a means to incorporate the following model components:

- Definition of input, output, and local signals (variables), including how one can be constructed mathematically from other signals and functions
- Definition of breakpoint vectors (can be shared between tables)
- Definition of dependent data tables (can be shared between functions); with no arbitrary limit to number of table dimensions
- Definition of table-based nonlinear functions
- Checkcase (verification) data

Through use of these elements, the schema allows a single text-based data file that sufficiently describes any arbitrary mapping between input values and output values. This capability is more than adequate to describe the most complex static aero model; other static components of a typical aerospace vehicle simulation can be described as well, including mass and inertia models and static thrust models.

Additional information that is or can be encoded in the model:

- Provenance, or pedigree, of the model (author, links to documentation like formal reports)
- Uncertainty boundaries (useful for performing Monte Carlo simulations)
- Record of changes made with granularity as fine as per data point

The full Document Type Definition is available from the project website; see Resources below.

3.3 The tools available thus far

The development of a standard for sharing data is a bit of a chicken-versus-egg scenario: until sufficient tools exist to make use of the exchange format, the format will not be useful; until the format is useful, tools will be slow to develop. To jump-start this process, several organizations have begun development of tools to help promote the use of the schema.

3.3.1 DAVEtools

A Java-based package, DAVEtools, has been developed at NASA Langley and provides two capabilities. First, it allows for a command-line exercise of the model (the user is prompted for input values and the output values are then calculated. Second, DAVEtools supports conversion of the model into Simulink®, a graphical modeling and analysis format developed by The Mathworks, Inc. as an adjunct to their Matlab® product. DAVEtools takes as input a model expressed in the XML-based grammar and produces a Simulink® .mdl file and associated setup.m script that can be opened and further developed within the Matlab Simulink environment. A verification script can be produced that will automatically verify the proper implementation in Simulink by comparing output vectors for several predefined input vectors to the model. This tool, including source code, is available upon request (see the Resources section of this paper).

3.3.2 Janus

A C++ application programming library, Janus, has been developed for the Australian Defense Science & Technology Organization (DSTO) that provides run-time loading and interpretation of a model expressed with the XML-based grammar; public release is pending.

3.3.3 NASA Ames Function Table Processor scripts

NASA Ames, an early innovator in digital flight simulation, has developed Perl-based import scripts that convert the tables found in the XML-based grammar into their Function Table Processor source format. The scripts also generate FORTRAN language code excerpts that represent the interpolation and buildup equations based on the tabular data. Availability of these scripts is internal to NASA Ames simulation projects.

3.3.4 Extensible Stylesheet (XSL) conversion

An XSL conversion stylesheet that documents a model expressed in the XML-based grammar into a Hypertext Markup Language (HTML) format, suitable for viewing with a conventional web browser, has been developed and is available (see the Resources section of this paper). It is named DAVE_html.xsl and serves as an example of how XML

3.4 Simple example

To illuminate what a simple aerodynamic function might look like when expressed in this XML-based grammar, an example is given below; this expression corresponds with the curve shown in figure 3.1:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DAVEfunc SYSTEM "DAVEfunc.dtd">
<DAVEfunc>
   <variableDef
      varID="angleOfAttack_d"
      name="Alpha" units="deg"
   />

   <variableDef
      varID="CmAlfa"
      name="Cma"
      units=""
   />

   <breakpointDef
      bpID="angleOfAttack_d_bp1">
      <bpVals>
   0, 10, 18, 20, 22, 23, 25, 27, 30
      </bpVals>
   </breakpointDef>

   <griddedTableDef gtID="CmAlfa_Table1">
      <breakpointRefs>
         <bpRef
            bpID="angleOfAttack_d_bp1"/>
      </breakpointRefs>
      <dataTable>
   -0.3, -0.2, -0.1, -.08, -0.05, -0.05,
   -0.07, -0.15, -0.6
      </dataTable>
   </griddedTableDef>

   <function name="Cm_alpha_func">
      <independentVarRef
      varID="angleOfAttack_d"/>
      <dependentVarRef varID="CmAlfa"/>
      <functionDefn>
         <griddedTableRef
            gtID="CmAlfa_Table1"/>
      </functionDefn>
   </function>

</DAVEfunc>
```

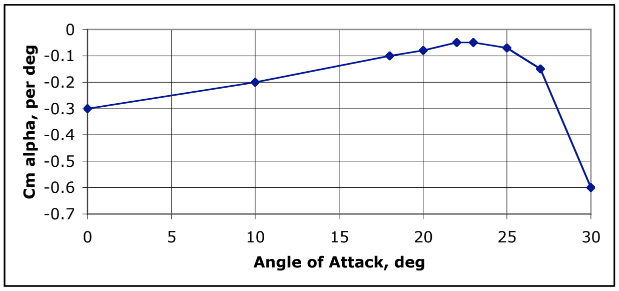

Figure 3.1 A simple aerodynamic function

This example shows an exceedingly simple aerodynamic model with a single input, angle of attack, and single interpolated function table giving the coefficient of pitch stability ($dC_m/d\alpha$) as a function of angle of attack. The two variables are defined (one for the input, `Alpha`, and one for the output, Cma. The input variable is expected to be in units of degrees; the output variable is dimensionless. The defined function is a one-dimension table with nine breakpoints and nine corresponding values. Behavior beyond the bounds of the valid table data (when Alpha is less than 0 or more than 90 degrees) is not defined; it is possible to specify whether to extrapolate or hold last value at either boundary. Likewise, the provenance, uncertainty, and checkcase information has been omitted for conciseness.

More elaborate examples are available at the project website (see the Resources section below).

4. Overview of the HDF5 Validation Data Standard

Development of a full-fledged exchange standard for simulation models will require checkcase data for dynamic models. While the current, limited, XML-based grammar does include checkcase data sufficient to verify the static models that can be encoded, this capability needs to be expanded to allow for the inclusion of dynamics in both the encoded model and in the checkcase data. Thus, a time history of the behavior of inputs, states, and outputs is necessary.

The features desirable in a standard time-history data file format include

- Based on an international data encoding standard
- Availability of tools to manipulate and transform the time-history data
- Compact (binary) yet computer-platform independent
- Tailorable to specific vehicle models

Hierarchical Data Format, version 5 (HDF5) meets those needs and has been selected as the encoding scheme for time-history checkcase data to verify dynamic flight vehicle model exchanges.

HDF was developed by the National Center for Supercomputing Applications (NCSA) (the birthplace of the original Netscape web browser) at University of Illinois at Urbana-Champaign (UIUC), along with significant contributions by others. The fifth version, HDF5, is actively supported and many "tools and I/O libraries" have been developed "for analyzing, visualizing, and converting scientific data." These tools and libraries are, for the most part, freely available under license from UIUC.

4.1 Existing JSF standard for flight test data

The US Department of Defense has adopted the HDF5 as the standard encoding for flight test data from the Joint Strike Fighter (F-35) program. This means the primary contractor (Lockheed-Martin) and participating government agencies and labs have the capability to read and write HDF5. As a result, a number of tools already exist that will benefit the standard model exchange effort.

The existing JSF HDF5 data standard, known as H5TD, supports encoding of both time-dependent and frequency-dependent data, so verification of exchanged flight models can be performed in both the time- and frequency-domains, an excellent recommended practice for simulation validation.

4.2 Tailorable

Since the source code for the NCSA HDF5 tools are available, portions of the software can be reused in specialized tools designed to assist in adoption of the model exchange standard.

4.3 Relationship to XML

HDF5 is, as the name implies, hierarchical in nature. This is similar to how information is organized in XML. Since the model structure itself is expressed using XML, it is convenient that the capability exists to translate time-history data from the compact HDF5 format into XML (human-readable) via the h5dump utility and back again using the h5gen program, both available from the NCSU HDF project website (see Resources).

5. Resources

Additional examples and links to related information, including the formal Document Type Definition and available utility software tools discussed in this paper, can be found at http://daveml.nasa.gov. There is link there to provide feedback and to subscribe to a discussion group, simstds@larc.nasa.gov.

Information about HDF5 is available from http://hdf.ncsa.uiuc.edu/HDF5; the documentation page (http://hdf.ncsa.uiuc.edu/HDF5/doc) is especially helpful.

6. Summary

The first version of the standard is fully developed and includes data standards for both modeling and validation data for dynamic models. Conceptually, the standard applies to any system dynamic model and validation of any system using time or frequency response data. It was specifically designed for aerodynamic models and it has a very well developed and tested XML format for function table data with a Math ML link for simple mathematical algebraic equations.

It is important to understand that since this is an interchange standard, it additionally has variable definitions and axis system definitions to facilitate the interchange of information. These components of the standard are not to require people to use certain variable names or axis systems, but to make it easy to communicate information via standard names and standard axis systems when desired.

The standard leverages existing computer standards of XML/MathML for the model and HDF-5 for transmittal of time history data (or frequency domain data) for the validation. The XML model definition has been tested in an exchange of a high performance fighter model between the Navy and NASA [6]. This testing of the standard was used to optimize the information content and format of the standard. Tools for importing and exporting to Simulink and Fortran models have been developed.

The standard has been formally adopted by the DSTO (Defense Science Technology Organization) of Australia as a model standard. This has become a format that they maintain all their model databases in. This is well beyond the concept of an interchange standard but shows the validity of the standard for representing aircraft math models.

The standard is well along in the ANSI standard application process with the American Institute of Aeronautics and Astronautics. The authors solicit feedback and interest on the standard.

7. References

[1] http://www.aiaa.org

[2] Jackson, E. Bruce and Hildreth, Bruce L.: "Flight Dynamic Model Exchange Using XML," AIAA paper 2002-4482, August 2002.

[3] Anon.: "*Recommended Practice, Atmospheric and Space Flight Vehicle Coordinates Systems*," ANSI/AIAA R-004-1992.

[4] Anon.: "*Standard for Distributed Interactive Simulation-Application Protocols*, Version 2.0, Fourth Draft (Revised)," IST-CR-94-50, March 1994.

[5] AIAA Simulation Standards Working Group: "*Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference*, Version 1.7b1," February 2004, available from http://daveml.nasa.gov/DTDs/1p7b1/DAVE-ML_ref.pdf

[6] Jackson, E. Bruce; Hildreth, Bruce L.; York, Brent W.; and Cleveland, William B.: "Evaluation of a Candidate Flight Dynamics Model Simulation Standard Exchange Format," AIAA paper 2004-5038, Providence, RI, August 2004.

Author Biographies

BRUCE HILDRETH is the older of the two Bruces and has thus earned the coveted #1 designation. He is Vice President and a Technical Fellow at SAIC in Lexington Park, Maryland. He has been involved in flight test, flight simulation and flight software and hardware development for over 30 years, and wrote the first physics-based flight simulation model at the Naval Air Warfare Center's Manned Flight Simulator (MFS) facility at Patuxent River, Maryland, of which he was the co-developer. He presently serves as Subcommittee Chair for Standards of the AIAA Modeling and Simulation Technical Committee.

BRUCE JACKSON is a senior research engineer at NASA Langley Research Center, Hampton, Virginia. He has been needlessly rehosting perfectly good simulation models for 24 years and would like to not have to do that again. He has written two complete flight simulation frameworks and is working on a third. He is lead guidance and controls engineer for the HL-20 lifting body concept and has designed or analyzed control laws for Pegasus, X-43A, X-48A, X-37 and blended-wing body transport aircraft. He has been involved in flying qualities investigations for supersonic transports including the Tupolev Tu-144 as well as conceptual models of the High Speed Civil Transport.