# Status of the AIAA Modeling and Simulation Format Standard

*E. Bruce Jackson, NASA Langley Research Center, Hampton, Virginia, USA*
*Bruce L. Hildreth, SAIC, Lexington Park, Maryland, USA*

## Synopsis

Aircraft flight dynamic models are developed in many different source formats. When an aircraft simulation was used entirely in-house, this diversity was not a problem. In recent years, however, a single stovepipe entity taking an aircraft from concept to production is rare. Instead, teaming arrangements with other manufacturing concerns and government agencies are now the norm. As a result, the sharing of aircraft flight dynamic models is much more commonplace. Unfortunately, each organization has their own simulation framework, coding standard, source language, and data format, which requires a laborious and error-prone manual translation or re-hosting of the simulation model between team members. To alleviate this time-consuming re-hosting, the American Institute of Aeronautics and Astronautics (AIAA) Modeling and Simulation Technical Committee (MSTC) has developed a text-based flight dynamic model exchange format, which represents a 'neutral ground' standard that is facility- and programming-language-independent while providing distinct advantages over current practices. The cost and time required to share and update aerodynamics, control laws, mass and inertia, and other flight dynamic components of the equations of motion of an aircraft or spacecraft simulation can thereby be reduced. A 2002 paper[1] estimated over $6 million in savings could be realized for one military aircraft type alone. This paper describes the result of efforts by the MSTC to develop a standard flight dynamic model exchange standard based on a specialized extensible markup language grammar. This standard, the Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML), is now being proposed as an ANSI/ISO standard.

## Motivation and Background

At the dawn of flight simulation, different aeronautical organizations and research laboratories developed increasingly sophisticated mathematical models of aircraft flight dynamics. Often these models were written as differential equations on early electronic analog computers. As the power of digital computers grew, the ability to numerically integrate these equations of motion allowed for solution of the resulting description of motion in real-time. FORTRAN was an early language of choice for flight simulation dynamic models, but due to computational limitations, much of the simulation calculations were written in machine language or assembler code. These usually included performing linear interpolation of wind-tunnel-derived aerodynamic coefficients. Only in the last few decades have flight models written entirely in higher-order languages been solvable in real-time on typical consumer-level computers.

Due to the diversity of early flight research (including airframe manufacturers, airline operators, research laboratories, flight training device manufacturers and military services), no single approach to writing flight simulation code emerged. Even today there are few agreed-upon standards for encoding flight dynamic mathematical models.

The importance of these mathematical models continues to grow, however. Due to the exponential increase in the cost of developing a new aerospace vehicle, the importance of using simulations for preflight prediction of flight performance, dynamic capabilities and flying qualities has become paramount. Program go-ahead decisions are often based on simulation and analytical results. At the same time, the sheer financial investment and risk of major aerospace acquisition programs has resulted in now-familiar teaming arrangements between the customer and one or more aircraft vendors, resulting in the need to share these flight simulation models reliably between dissimilar simulation laboratories. Simultaneously, the arrival of the electronic frontier in the form of the Internet has made the reliable transmission of large amounts of data, such as a flight simulation model, even more convenient - nine-track tapes are now in museums.

---

[1] Jackson, E. Bruce and Hildreth, Bruce L.: **Flight Dynamic Model Exchange using XML**. AIAA Paper 2002-4482

So we find ourselves at a place in history in which the reliable and effortless exchange of a flight simulation model between various developers, analysts, testers, students and other users is not only extremely important but should be effortless to accomplish. This is not the case because one piece of the puzzle is missing: there is no agreed-upon standard for describing, digitally, in an unambiguous way, the flight dynamics of an aircraft.

Two examples illustrate this problem: what should we call angle-of-attack, and how should tables of aerodynamic coefficients be described?

In Western engineering texts, the Greek letter $\alpha$ is used to denote the relative angle between the airframe and the relative velocity of the aircraft through the atmosphere. Within NASA simulations, however, such variable names such as `ALPDEG`, `ALFA`, and `angleOfAttack_deg` have been used. Since no agreement exists on how to represent $\alpha$ in a Roman alphabet variable name (although we can agree on the symbol), not to mention the confusion over angular units of radians or degrees (is `ALFA` in degrees, or radians?), an impediment between rapid digital model exchange is obvious. Indeed, the fact that several names are used gives rise to the opportunity for them to have different definitions, which is a further impediment to correctly exchanging information.

The vast bulk of an aerodynamic model in a high-fidelity flight simulation is taken up by large, multi-dimensional tables of coefficients that represent non-linear relationships, or functions, between increments of forces and moments in response to changes in an aircraft's flight condition - Mach number, angle-of-attack, sideslip angle, control deflection, and throttle setting, to give a common example. Since linear function interpolation was a difficult problem to solve in early digital simulations (due to the requirement for a floating-point division, which used to take many more cycles than a multiplication), as many ways to represent interpolation tables exist as there are flight research organizations. Thus, the exchange of an aerodynamic model between simulation facilities almost always requires the reformatting of such tables, usually a lengthy process. This sometimes can be automated but, since there is no agreed 'standard' format, the automation algorithm must be created and tested for each such new exchange, while simultaneously converting variable names and sometimes rewriting the associated aerodynamic build-up equations to match the new facility's conventions. The resulting new implementation must be validated once it's completed as well. This 'rehosting' of a perfectly fine flight simulation model, when it changes hands between aeronautical partners, can take months.

We don't have to look far to find help, however; the World-Wide Web consortium[2] has developed a means to describe data sets - XML, the eXtensible Markup Language[3] - and mathematical relationships - MathML, the Mathematics Markup Language[4] (itself an extension of XML).

## Objective

The AIAA, through its MSTC, has undertaken the task of developing standards for the exchange of flight simulation models, primarily through the application of XML and MathML technology and through establishment of a standard set of common flight simulation variable names. This provides a way to unambiguously express, through a facility- and programming-language-neutral representation, the atmospheric flight dynamics of an aerospace craft.

By development of two enabling technologies:

- definition of a standard set of variable names and axis systems common to flight simulation domain, and
- development of an 'application grammar' of XML unique to flight dynamics modeling,

it is possible to describe the complete aerodynamic model of a flight vehicle in a human- and machine-readable way. This model can then be used in a real-time piloted simulation, a flight performance analysis tool, a Monte Carlo flight trajectory trade study, or in flying qualities estimation

[2] http://www.w3.org
[3] http://www.w3.org/XML
[4] http://www.w3.org/Math

methods. The format is also suitable for archival purposes since it can include provenance information and extensive reference documentation.

The Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML)[5] can encode:

- The aerodynamic model 'build-up' equations, or mathematical model
- Multi-dimensional non-linear tables of coefficients (or equivalent polynomial describing functions)
- Statistical and uncertainty parametric variations in the model
- Provenance or history of various features of the model
- Checkcase data for verification of the model

## Benefits of an XML-based Standard

Typically there are many simulations of a specific aircraft model. There are:

- Prime manufacturer developmental simulations (manned and unmanned)
- Military research, development, test and evaluation (RDT&E) simulations (manned and unmanned)
- Government research laboratory simulations (manned and unmanned)
- Customer country simulations (manned and unmanned)
- Part task crew training simulations
- Full fidelity crew training simulations
- Crew training mission capable simulations

The principle benefit of the standard is to improve collaboration among these simulators and simulation facilities. Typically there are many simulations of the same system, often numbering over 100. Also typically, these simulations only cross-pollinate model improvements and corrections on an ad hoc basis, if at all; thus, improvements made to one simulation do not normally proliferate to the other simulations. A barrier to cooperation is that each simulation generally has different hardware and software architectures.

Since it is virtually impossible to have government labs and industry standardize all simulation architectures (also probably a bad idea for reasons beyond the scope of this paper), the MSTC arrived at the concept of a standard exchange format. If each of the simulation teams adopted this exchange format they would simply have to support one set of import and export applications (see figure 1) to exchange simulation models between any other simulation team. Practically speaking, each of these teams must already create export and import routines to exchange models and data with other interested parties. The standard just creates one format for all to use. This would save significant effort. Most importantly, this saving of effort would facilitate collaboration, which is the true goal and should provide the greatest benefit.

[5] **Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference**, version 2.0RC1, Oct. 2007. Available from http://daveml.nasa.gov/DTDs/2pRC1/DAVE-ML_ref.pdf

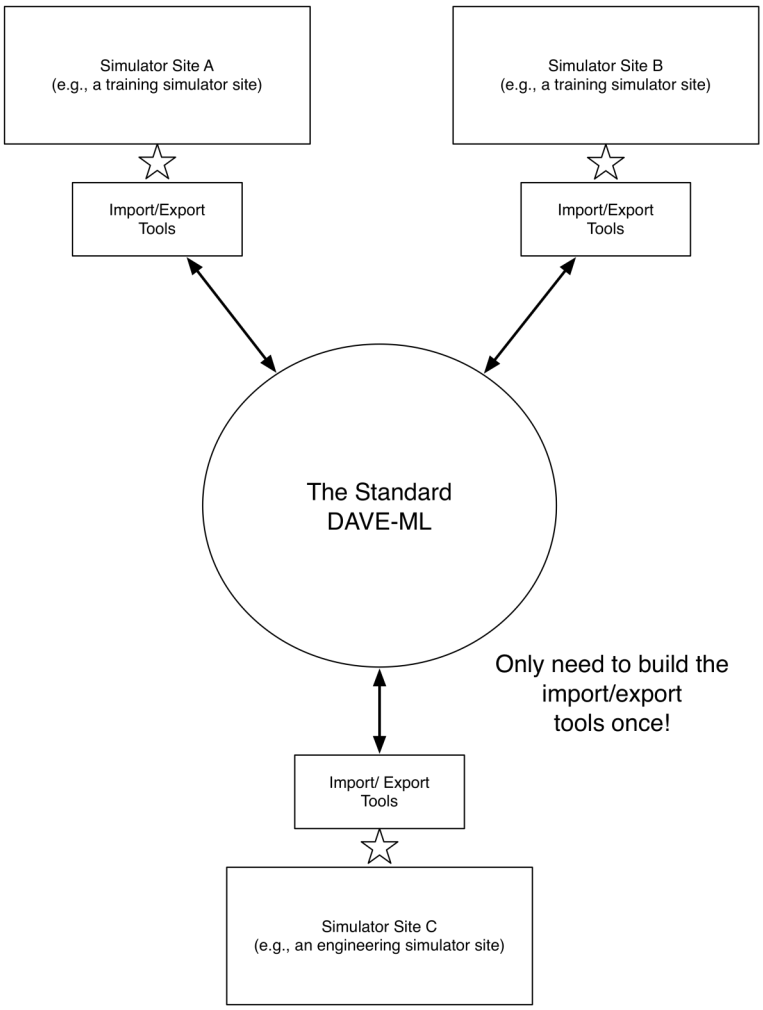


**Figure 1. Schematic of improved simulation model exchange using a standard format**

### *General Benefits*

The general benefits of the standard, which apply to any type of simulation, include:

| Feature | Benefit |
|---|---|
| Improved standard function table format | 1. Write and verify import/export routines only once<br>2. No requirement to rewrite or re-host any function tables |
| Function table format includes data provenance | 3. Provenance allows the user to track where the data came from and its intended use<br>4. Reduces mistakes in using functions improperly or for the wrong flight regime or configuration |
| The standard is simulation architecture and code neutral | 5. Works with virtually any facility and language |
| Implemented in XML | 6. Many off the shelf editors and tools available facilitate the development of the import/export tools<br>7. Resulting file both human and computer readable |
| Equations can be represented in MATH-ML | 8. No requirement to rewrite parts of the aero model for many aero model changes<br>9. Fewer mistakes when updating a model |
| Standard includes use of standard axis systems | 10. No requirement to define the axis systems used in the model<br>11. Fewer mistakes when transferring model |

| Feature | Benefit |
|---|---|
| Standard variable names | 12. Simplifies transfer of information between facilities; use of standard names that already have clear definitions eliminates requirement to document the variables<br>13. Custom or new variables are allowed; however the user must clearly define them, including units and axis system if applicable |
| Standard naming convention for model-specific variables | 14. Allows the user community to expand the standard to further simplify the exchange of information |
| Naming convention includes designation of states, state derivatives and (optionally) controls | 15. Improves documentation clarity<br>16. Key variables are easy to find<br>17. Facilitates software maintenance |
| Naming convention includes units as part of the variable name | 18. Improves documentation clarity-reduces the occurrence of errors due to improper mixing of units<br>19. Facilitates software maintenance |

### *Impact to the Life Cycle Support of the Simulation*

The benefits of the standard can be best expanded upon by discussing how the standard would apply in the life cycle of a simulation system. The phases of the engineering development and life cycle support a simulator that are clearly benefitted by the standard are listed below and discussed in the ensuing paragraphs.

- **Initial simulator development**
  - Design
  - Code
  - Test
  - Verification
- **Life cycle support**
  - Software maintenance
  - Update/ modify/ technology refresh

**Initial simulator development - Design**

One of the largest efforts in the design phase of a simulator model development is locating the data required for the model. For aerodynamic models this includes the describing functions that represent the forces and moments on the airplane. The same is true with engine models whose functions represent the thrust and fuel consumption. Once these functions are designed, the standard provides the ability to include the provenance regarding each function as part of the data. This is critically important in the development of a model. Where did this data come from? What is its intended use? What is the configuration of the aircraft for this data? This is all critical information provided by the standard that is not typically available in other data table representations.

While the technical benefits of the standard are clear, probably the greatest benefit of widespread use of the standard is that the models will be more accessible and will improve the collaboration between facilities and developers. This collaboration will increase and make models better, lowering the cost of development of a new simulation.

**Initial simulator development - Code**

When coding a new model the standard is a great benefit if the convention for variable naming is followed. Key to the variable naming is the tying of the axis system, measurement units, and designation of states to the variable.

Starting with the latter, the standard variable naming convention designates which variables are states, state derivatives, or (optionally) controls. The dynamics of the simulation, which are clearly the hardest aspects of an aircraft model to match to criteria data, are all controlled by the states and state

derivatives. Using a standard naming convention makes it easy to track which variable are “key” in the simulation.

While initially controversial, the standard variable naming convention uses units as a suffix. Initially developers complained about this due to the additional typing that was required. However as they used this feature the complaints dissolved, as it reduced mistakes caused by mixing of improper units.

**Initial simulator development - Test**

At its present level of development the standard supports the automated ability to check the mapping between model inputs and outputs. This is by itself a valuable capability. It also provides, for certain check cases, the sharing of internal or intermediate variable values for debugging purposes.

**Initial simulator development - Verification**

The authors define *verification* as assuring that the model is performing as designed. The authors define *validation* as assuring that the model realistically represents the vehicle it is simulating.

The standard helps simplify the verification process by standardizing the import and export process, reducing the probability of make a mistake in the importing of new model data. Also the capability to include static check data is a valuable verification check. However, the standard has no method to ensure that the function data table, while imported accurately, represents the aircraft or weapon system in question; it only helps reduce the probability that a mistake was made in implementing said data. Future versions of the standard will include the ability to exchange time history data for the purpose of dynamic verification of the model after the implementation.

**Life Cycle Support - Software Maintenance**

The biggest problem with software maintenance is that it is often performed years after the development of the simulation when the original developers are no longer available. The new maintainers must unravel the code to understand properly how to make a modification or improvement. The aforementioned variable naming convention dramatically simplifies this by identifying the states, state derivatives, controls and the units of the variables. States and state derivatives are the key variables that determine the dynamics of the model. The units in the variable name help the self-documenting features of the code and help prevent making a modification using the wrong units. These two simple conventions will simplify maintenance and should be quickly accepted by maintainers.

Often maintenance is required to update the simulation model's function tables to reflect changes to the model, such as extending the flight envelope that the simulation can represent or updating the engine tables to reflect upgraded engines installed in the airplane. The use of a standard function table representation makes incorporation of these new tables almost trivial.

**Life Cycle Support - Updates / Technology Refresh**

While the standard itself does not aid in upgrading the computers or upgrading the software language (two typical examples of technology refresh), the use of a standard can simplify this process. No one can predict what the future will bring, but the more standardization used in the development of a simulator, the greater the likelihood that there will be tools to help implement the model in the next level of technology (computer hardware and software). In other words, the more unique the software is, the harder it is to adopt new technology. The more standardized a system is, the easier it is to keep technology current because there will be a larger community of practicioners working with these tools and supporting the standard. This is a significant advantage when trying to update the hardware and software of the simulator.

## *Key Benefit: Improved Model Fidelity/Transfer of Training*

The above are primarily cost and schedule benefits. Perhaps the biggest technical benefit of the standard is that improved collaboration between all the different types of simulations results in an improved math model on the trainer. For example, one facility may be concentrating on high angle-

of-attack and have developed an improved high angle-of-attack database. Others may have found an error in the flaps-down approach-to-land database. Another simulation may be developing engine or stores models and have improved the engine tables or the aerodynamic effects of stores, etc. With the barriers to data transfers that exist now it is rare that these improvements get widely distributed. The application of a standard significantly reduces or eliminates these barriers and will allow all these improvements to be more easily distributed throughout the simulation community with little cost or effort. Therefore the simulation will now have a larger flight envelope with improved utility and better fidelity.

### *Cost Benefit*

Those familiar with the translation of models from one simulation architecture to another can easily appreciate the potential for savings created by this standard. The authors of this paper earlier estimated (see reference 1) an annual potential savings of $6.8M (USD) in a case study of simulators for one military aircraft. This case study included 59 simulators (now well over 100 for this one aircraft model). Thirty of the 59 simulations were pilot training simulations, the rest were RDT&E or aircraft manufacturer simulations. This paper also claims that the biggest potential benefit is from increased collaboration between the people working on the different simulations. It is difficult to quantify the true gains in time and money, but the authors take the position that the leverage of improved collaboration is huge. A small gain in collaboration results in a great benefit to the simulation community.

## Example DAVE-ML models

Some examples of the ability of the DAVE-ML grammar to depict aspects of typical engineering flight simulation aerodynamic models are shown below; more examples can be found on the DAVE-ML website (http://daveml.nasa.gov/examples.html).

### Minimal one-dimensional table

The example below is the minimal realization of the single, one-dimensional representation of a lift coefficient vs. angle-of-attack relationship. This is a valid DAVE-ML model, but it doesn't include desirable information about the source, purpose, or any modifications (typical of many legacy models). Figure 2 depicts the information contained in the example.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DAVEfunc
  PUBLIC "-//NASA//DTD for Flight Dynamic
  Models - Functions 2.0//EN"
  "DAVEfunc.dtd">
<DAVEfunc>
  <fileHeader>
    <author name="Bruce Jackson"
            org="NASA"/>
    <creationDate date="2002-03-11"/>
  </fileHeader>
  <variableDef name="alpha" varID="alpdeg"
               units="deg"/>
  <variableDef name="CL" varID="cl"
               units=""/>
  <function name="CL">
    <independentVarPts varID="alpdeg">
      -4.0,0.,4.0,8.0,12.0,16.0
    </independentVarPts>
    <dependentVarPts varID="cl">
      0.0,0.2,0.4,0.8,1.0,1.2
    </dependentVarPts>
  </function>
</DAVEfunc>
```

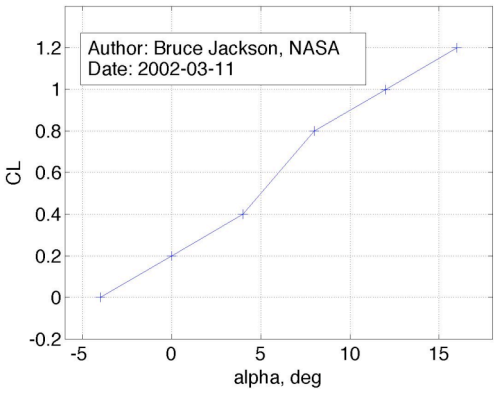


**Figure 2. Depiction of the one-dimensional function found in the first two examples**

### A more complete one-dimensional table

A more realistic encoding of the same lift coefficient vs. angle-of-attack function, depicted in figure 2, is given below. This implementation contains more useful information about the model including a description and the modification history. While whitespace is optional for most XML grammars,

human readability warrants judicious use of comments and whitespace. This example is not really a useful aerodynamic model since it only describes one function; most models will contain dozens if not hundreds, of functions as well as verification data, buildup equations and uncertainty distributions.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DAVEfunc PUBLIC
          "-//NASA//DTD for Flight Dynamic Models - Functions 2.0//EN" "DAVEfunc.dtd">
<DAVEfunc>
  <fileHeader>
    <author name="Bruce Jackson" org="NASA Langley Research Center"
            email="e.b.jackson@nasa.gov"/>
    <creationDate date="2002-03-11"/>
    <fileVersion>$Revision: 286 $</fileVersion><description>
        Coefficient of lift (non-dimensional) versus angle-of-attack, deg.
        Example file for DAVE function table format. This example is the simplest version.
    </description>
    <modificationRecord modID="A" date="2002-03-11">
      <author name="Bruce Jackson" org="NASA Langley Research Center"
              email="e.b.jackson@nasa.gov"/>
      <description>
        Added varID to dependentVarPts and independentVarPts, per DTD 1.5b2 change.
        Also changed author's xns address to e-mail address.
      </description>
    </modificationRecord>
    <modificationRecord modID="B" date="2006-11-17">
      <author name="Bruce Jackson" org="NASA Langley Research Center"
              email="e.b.jackson@nasa.gov"/>
      <description>
        Added date to modificationRecord per DTD 1.9 change.
      </description>
    </modificationRecord>
  </fileHeader>
  <variableDef name="alpha" varID="alpdeg" units="deg"/>
  <variableDef name="CL" varID="cl" units=""/>
  <function name="CL">
    <independentVarPts varID="alpdeg">
      -4.0, 0., 4.0, 8.0, 12.0, 16.0
    </independentVarPts>
    <dependentVarPts varID="cl">
      0.0, 0.2, 0.4, 0.8, 1.0, 1.2
    </dependentVarPts>
  </function>
</DAVEfunc>
```

**A more complex aero model with a two-dimensional function**

The example shown below begins to approach a useful, albeit fictional, model. It describes an aerodynamic model composed of a single two-dimensional coefficient of lift as a function of angle-of-attack and Mach number; as is more typical of higher fidelity models, the set of independent values for the orthogonal table are declared separately for potential reuse with other similar-dimensioned tables (although this example doesn't demonstrate table reuse).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DAVEfunc
          PUBLIC "-//NASA//DTD for Flight Dynamic Models - Functions 2.0//EN"
          "DAVEfunc.dtd">
<DAVEfunc>
  <fileHeader>
    <author name="Bruce Jackson" org="NASA Langley Research Center"
            email="e.b.jackson@nasa.gov"/>
    <creationDate date="2002-03-01"/>
    <fileVersion>$Revision: 287 $</fileVersion>
    <description> Coefficient of lift (non-dimensional) - single two-dimensional table.
      Example file for depiction of typical DAVE function table format.
      This example is more complex.
    </description>

    <reference refID="REF01" author="Raney, David L." title="F-19A Basic Aerodynamics Model"
      accession="NASA TM-4302" date="1992-07-31"/>
    <reference refID="REF02" author="Buggati, Richard W."
      title="F-19A Rotary Aerodynamics Model"
      accession="NASA TM-4303" date="1993-07-31"
      xlink:href="http://dcb.larc.nasa.gov/models/tm4303.pdf"/>
    <reference refID="REF03" author="Aviation Leak &amp;amp; Space Tautology"
```

```
      title="F-19A Falling Short on Design Performance Numbers"
      accession="Vol. 45, Number 12"
      date="1993-07-03"/>

    <modificationRecord modID="A" refID="REF03" date="2002-03-01">
      <author name="Hildrup, Bruce L." org="SAID Patuxent River">
        <address>
          1087 Exploration Parkway Ste 201
          Lexington Park, MD 20654
        </address>
      </author>
      <description> Reversed sign on drag term for better performance! </description>
    </modificationRecord>

    <modificationRecord modID="B" date="2006-11-16">
      <author name="Bruce Jackson" org="NASA Langley Research Center"
              email="bruce.jackson@nasa.gov">
        <address>
          MS308 NASA
          Hampton VA 23681 USA
        </address>
      </author>
      <description>
        Renamed from .xml to .dml; changed to conform to DAVEfunc.dtd 1.9: added 'date'
        to modification record, renamed docID attribute in documentRefs to refID;
        corrected date formats to ISO; added fileVersion element with
        Revision keyword.
      </description>
    </modificationRecord>

    <modificationRecord modID="C" date="2008-02-27">
      <author name="Bruce Jackson" org="NASA Langley Research Center"
              email="bruce.jackson@nasa.gov">
        <address>
          MS308 NASA
          Hampton VA 23681 USA
        </address>
      </author>
      <description>
        Updated to 2.0 (fileCreationDate + functionCreationDate -> creationDate);
        updated email address
      </description>
    </modificationRecord>

</fileHeader>

  <!-- ================== -->
  <!--  Input variables   -->
  <!-- ================== -->

  <variableDef name="angleOfAttach_d" varID="ALPHA" units="deg" symbol="#x3B1">
    <description> Instantaneous true angle-of-attack, in degrees </description>
    <isStdAIAA/> <!-- this is a standard AIAA simulation variable -->
  </variableDef>

  <variableDef name="MachNumber" varID="MACH" units="ND" symbol="M">
    <isStdAIAA/> <!-- this is a standard AIAA simulation variable -->
  </variableDef>

  <!-- ================== -->
  <!--  Output variable   -->
  <!-- ================== -->

  <variableDef name="CL" varID="CL" units="ND" symbol="CL">
    <description> Coefficient of lift based on alpha and mach. </description>
  </variableDef>

  <!-- ================== -->
  <!--  Breakpoint values -->
  <!-- ================== -->

  <breakpointDef name="alpha_bp" bpID="ALPHA1" units="deg">
    <description> Alpha breakpoints for most basic aero data </description>
    <bpVals> -4.0, 0., 4.0, 8.0, 12.0, 16.0 </bpVals>
  </breakpointDef>

  <breakpointDef name="mach_bp" bpID="MACH1" units="ND">
```

```
    <bpVals> 0.0, 0.4, 0.8, 0.9, 0.95, 0.99, 1.00, 1.01, 1.05, 1.2 </bpVals>
  </breakpointDef>

  <!-- ================== -->
  <!--      Functions     -->
  <!-- ================== -->

  <function name="Basic CL">
    <description> Basic coefficient of lift table as a function of Mach and angle of attack
</description>
    <provenance>
      <author name="Bruce Jackson" org="NASA Langley Research Center" xns="@bjax"/>
      <creationDate date="Jul-1994"/>
      <documentRef refID="REF01"/>
      <documentRef refID="REF02"/>
      <modificationRef modID="A"/>
    </provenance>
    <independentVarRef varID="MACH" min="0.3" max="0.95" extrapolate="max"/>
    <!-- Mach breakpoints -->
    <independentVarRef varID="ALPHA" min="-0.4" max="16.0" extrapolate="both"/>
    <!-- Alpha breakpoints -->
    <dependentVarRef varID="CL"/>
    <functionDefn name="CL_FN">
      <griddedTable name="CL_TABLE">
        <breakpointRefs>
          <bpRef bpID="MACH1"/>
          <bpRef bpID="ALPHA1"/>
        </breakpointRefs>
        <dataTable>
          <!-- Note: last breakpoint (ALPHA1) changes most rapidly -->
 <!-- -4.0,        0.,      4.0,      8.0,       12.0,      16.0      alpha values  -->
   9.5013e-01 6.1543e-01 5.7891e-02 1.5274e-02 8.3812e-01 1.9343e-01 <!-- Mach 0.0  -->
   2.3114e-01 7.9194e-01 3.5287e-01 7.4679e-01 1.9640e-02 6.8222e-01 <!-- Mach 0.4  -->
   6.0684e-01 9.2181e-01 8.1317e-01 4.4510e-01 6.8128e-01 3.0276e-01 <!-- Mach 0.8  -->
   4.8598e-01 7.3821e-01 9.8613e-03 9.3181e-01 3.7948e-01 5.4167e-01 <!-- Mach 0.9  -->
   8.9130e-01 1.7627e-01 1.3889e-01 4.6599e-01 8.3180e-01 1.5087e-01 <!-- Mach 0.95 -->
   7.6210e-01 4.0571e-01 2.0277e-01 4.1865e-01 5.0281e-01 6.9790e-01 <!-- Mach 0.99 -->
   4.5647e-01 9.3547e-01 1.9872e-01 8.4622e-01 7.0947e-01 3.7837e-01 <!-- Mach 1.00 -->
   1.8504e-02 9.1690e-01 6.0379e-01 5.2515e-01 4.2889e-01 8.6001e-01 <!-- Mach 1.01 -->
   8.2141e-01 4.1027e-01 2.7219e-01 2.0265e-01 3.0462e-01 8.5366e-01 <!-- Mach 1.05 -->
   4.4470e-01 8.9365e-01 1.9881e-01 6.7214e-01 1.8965e-01 5.9356e-01 <!-- Mach 1.2  -->
        </dataTable>
      </griddedTable>
    </functionDefn>
  </function>
</DAVEfunc>
```

**An excerpt showing build-up calculations**

The example shown here is an excerpt from the HL-20 lifting body aerodynamic model, available from the DAVE-ML website[6]. It depicts the use of MathML-2 markup to describe how the outputs from various function descriptions (not shown) are combined through powers of angle-of-attack to form the lift coefficient contribution of the lower-left body flap.

This excerpt demonstrates the encoding of the equation

$$C_{L_{\delta BF_{LL}}} = C^0_{L_{\delta BF_{LL}}} + \alpha C^1_{L_{\delta BF_{LL}}} + \alpha^2 C^2_{L_{\delta BF_{LL}}} + \alpha^3 C^3_{L_{\delta BF_{LL}}}$$

Note that in this excerpt, the variables identified as `ALP`, `CLBFLL0`, `CLBFLL1`, `CLBFLL2`, and `CLBFLL3` have been defined elsewhere (`ALP` is angle of attack and is an input to the model; the other variables are outputs from multidimension function tables that are not shown).

This excerpt gives some indication of the method of describing aerodynamic buildup equations in a DAVE-ML model file.

```
<!--    lower left body flap lift buildup -->
  <variableDef name="CLdbfll" varID="CLBFLL" units="">
    <description>
        Lift contribution of lower left body flap deflection
```

[6] http://daveml.nasa.gov

```
      CLdbfll = CLdbfll_0 + alpha*(CLdbfll_1 + alpha*(CLdbfll_2
                          + alpha*CLdbfll_3))
  </description>
  <calculation>
    <math>
      <apply>
      <plus/>
        <ci>CLBFLL0</ci>
        <apply>
          <times/>
          <ci>ALP</ci>
          <apply>
            <plus/>
            <ci>CLBFLL1</ci>
            <apply>
              <times/>
              <ci>ALP</ci>
              <apply>
                <plus/>
                <ci>CLBFLL2</ci>
                <apply>
                  <times/>
                  <ci>ALP</ci>
                  <ci>CLBFLL3</ci>
                </apply> <!--            a*c3   -->
              </apply> <!--        (c2 + a*c3)  -->
            </apply> <!--        a*(c2 + a*c3)  -->
          </apply> <!--    (c1 + a*(c2 + a*c3)) -->
        </apply> <!--    a*(c1 + a*(c2 + a*c3)) -->
      </apply> <!-- c0 + a*(c1 + a*(c2 + a*c3)) -->
    </math>
  </calculation>
</variableDef>
```

## Project Status

### *Status of the draft Standard*

As of this writing (April 2008) the standard is in final editing by the American Institute of Aeronautics and Astronautics (AIAA), the largest aerospace professional society in the United States. It is being prepared as an American National Standards Institute (ANSI) standard by the Modeling and Simulation Technical Committee (MSTC) of the AIAA. After ANSI acceptance the DAVE-ML standard is planned to be submitted for consideration as an International Organization for Standardization (ISO) standard.

The "ANSI Flight Dynamic Model Exchange Standard, DAVE-ML" (the current working title) includes:

- Standard function table definition and convention
- Standard variable name definitions and convention
- Standard axis system definitions
- Standard static math equation representation.
- A reference manual, with examples, for the DAVE-ML grammar.

### *Existing Tools*

With the release of early DAVE-ML specifications, some preliminary tools to assist in using and importing DAVE-ML compliant simulation models have emerged. An up-to-date listing is presently available at the DAVE-ML website, http://daveml.nasa.gov/tools.html. At the time of this writing (April 2008) these include:

**DAVEtools**

This Java-based toolset provides two features at present: an interactive DAVE-ML model evaluation capability, and a means to realize a DAVE-ML model as a Simulink® block diagrams with support for most DAVE-ML functions. This tool is available from the DAVE-ML website[7] under the NASA Open Source software license.

**Janus**

This C++ library was developed by Australia's Defence Science and Technology Organisation (DSTO)[8]. It provides an application programming interface (API) that allows simulations to read and execute DAVE-ML models at run-time. It supports most DAVE-ML functions and some DSTO-unique extensions. It is available under the DSTO Open Source License from DSTO[9].

**DaveMLTranslator**

This C++ API was developed for use in NASA Langley Research Center's LaSRS++ real-time simulation framework[10]. It supports some DAVE-ML functions and is available upon written request under limited license from NASA Langley[11].

**XSLT**

Using an existing XML capability, extensible stylesheet translation technology (XLST), it is possible to automatically transform DAVE-ML files into HTML-based documentation. Stylesheets and examples can be found at the DAVE-ML website[12].

## Future Work

The next effort is to add a standard method for the exchange of dynamic verification data. Clearly this is needed, because after a model is exchanged between simulation facilities, the model must be fully verified. Any such exchange should include dynamic verification data. The present markup set only supports static (single time slice) verification data.

The MSTC has tentatively decided to adopt and tailor an “apparent” emerging flight test time history standard. This is a standard made for the Joint Strike Fighter (F-35) program. The simulation exchange standard would be a subset of the flight test data standard. It is implemented in Hierarchical Data Format 5 (HDF 5). HDF is a publically released architecture for storing large amounts of data of all types. Flight simulation applications are relatively simple compared to the capabilities of available HDF tools and products. HDF was started by the National Center for Supercomputing Applications and the University of Illinois at Urbana-Champaign[13]. It was not included in the present standard because of the difficulty of obtaining public release of specific flight-test related tools that should greatly facilitate the use HDF 5 in flight simulation applications. It is hoped that these issues will be shortly overcome and the next iteration of the standard will support exchange of dynamic validation and verification data.

After this minor hurdle is overcome, a more difficult task looms. How do we standardize on modeling dynamic systems? The proposed standard is limited to static systems, that is, models that are stateless. Additional syntax is needed to address initialization and integration methods, if dynamics are to be added to the standard. Control systems and engine models are the most obvious examples of dynamic models that need to be exchanged between simulation facilities.

---

[7] http://daveml.nasa.gov/DAVEtools.html
[8] Brian, Young, Newman, Curtin and Keating: **The Quest for a Unified Aircraft Dataset Format.** SIAA paper, August 2005
[9] http://www.dsto.defence.gov.au/research/4675
[10] Hill, Melissa A. and Jackson, E. Bruce: **The DaveMLTranslator: An Interface for DAVE-ML Aerodynamic Models.** AIAA Paper 2007-6890.
[11] mailto:bruce.jackson@nasa.gov
[12] http://daveml.nasa.gov/examples.html
[13] http://hdf.ncsa.uiuc.edu/HDF5/index.html

Ongoing discussions within the simulation standards community[14] indicate a desire to add matrix and vector definitions to the current DAVE-ML markup grammar. Initial proposals for this capability are presently being evaluated for inclusion into later versions of DAVE-ML.

Simulink®, by the The Mathworks, Inc., is a widely used tool in simulation of diverse dynamic systems. In standardizing the exchange of dynamic system models, the MSTC is considering whether this commercial product is the equivalent of a de-facto standard. Why create something that already exists and has wide acceptance? If DAVE-ML were to include dynamic systems as part of its repertoire, it might represent a non-proprietary encoding that captures some of the information contained in Simulink models. Informal discussions on this topic have taken place and will continue.

## Summary

The current draft AIAA Standard for flight simulation models represents an on-going effort to improve the productivity of practitioners of the art of digital flight simulation (one of the original digital computer applications). This initial release provides the capability for the efficient representation and exchange of an aerodynamic model in full fidelity; the DAVE-ML format can be easily imported (with development of site-specific import tools) in an unambiguous way with automatic verification. An attractive feature of the standard is the ability to coexist with existing legacy software or tools.

The draft Standard is currently limited in scope to static elements of dynamic flight simulations; however, these static elements represent the bulk of typical flight simulation mathematical models. It is already seeing application within U.S. and Australian government agencies in an effort to improve productivity and reduce model rehosting overhead. An existing tool allows import of DAVE-ML models into a popular simulation modeling and analysis tool, and other community-contributed tools and libraries can simplify the use of DAVE-ML compliant models at compile- or run-time of high-fidelity flight simulation.

[14] mailto:simstds@larc.nasa.gov