



J.F. Taylor, Inc.



Benefits to the Simulation Training Community of a New ANSI Standard for the Exchange of Aero Simulation Models

Unclassified

Bruce Hildreth
JFTI

E. Bruce Jackson
NASA Langley Research Center



Problem

- **Training simulator issues**
 - High development cost
 - Difficult and costly to maintain s/w
 - Difficult and costly to make s/w upgrades
- **Result**
 - Aircraft models lag behind the aircraft configuration
 - Aircraft models are not the best available

The fidelity of the model suffers -> crew training suffers

Many Simulations are Available to Solve the Problem

- Prime manufacturer's developmental simulations
- DoD RDT&E simulators
- NASA RDT&E simulators
- Customer Country simulators
- Part-task crew training simulators
- Full fidelity crew training simulators
- Mission training simulators

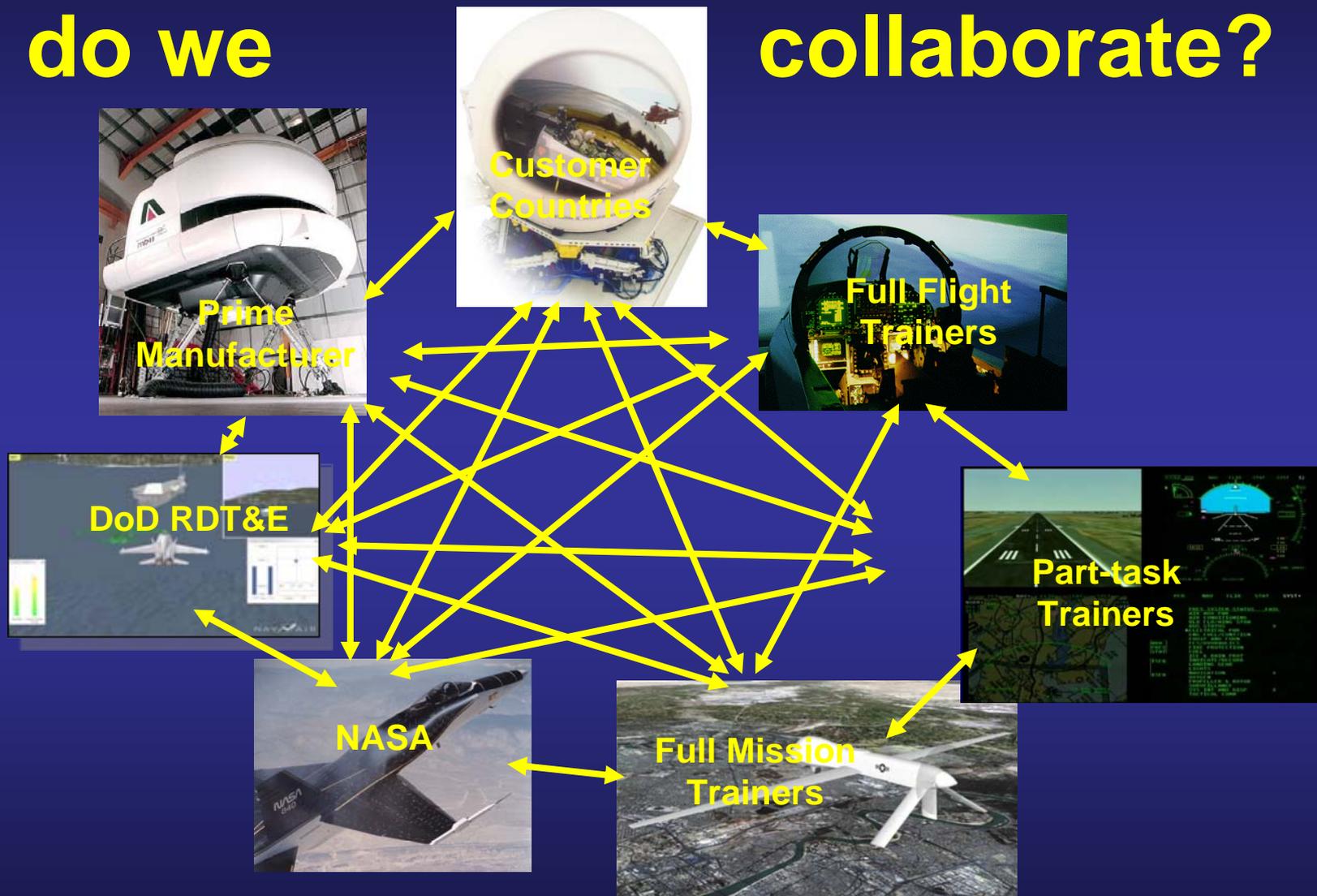
How can we lower the barriers to collaboration between these simulators?

My definition of collaboration- relative to training simulation

- Assuring what is learned at one simulator benefits another simulator
- Examples
 - Simulator A learns that a model aero function is wrong and corrects it
 - Simulator B changes it's model to validate it's Power Approach (PA) database
 - Simulator C adds a high angle of attack database
 - Simulator D adds higher actuator fidelity model to simulate an actuator malfunction

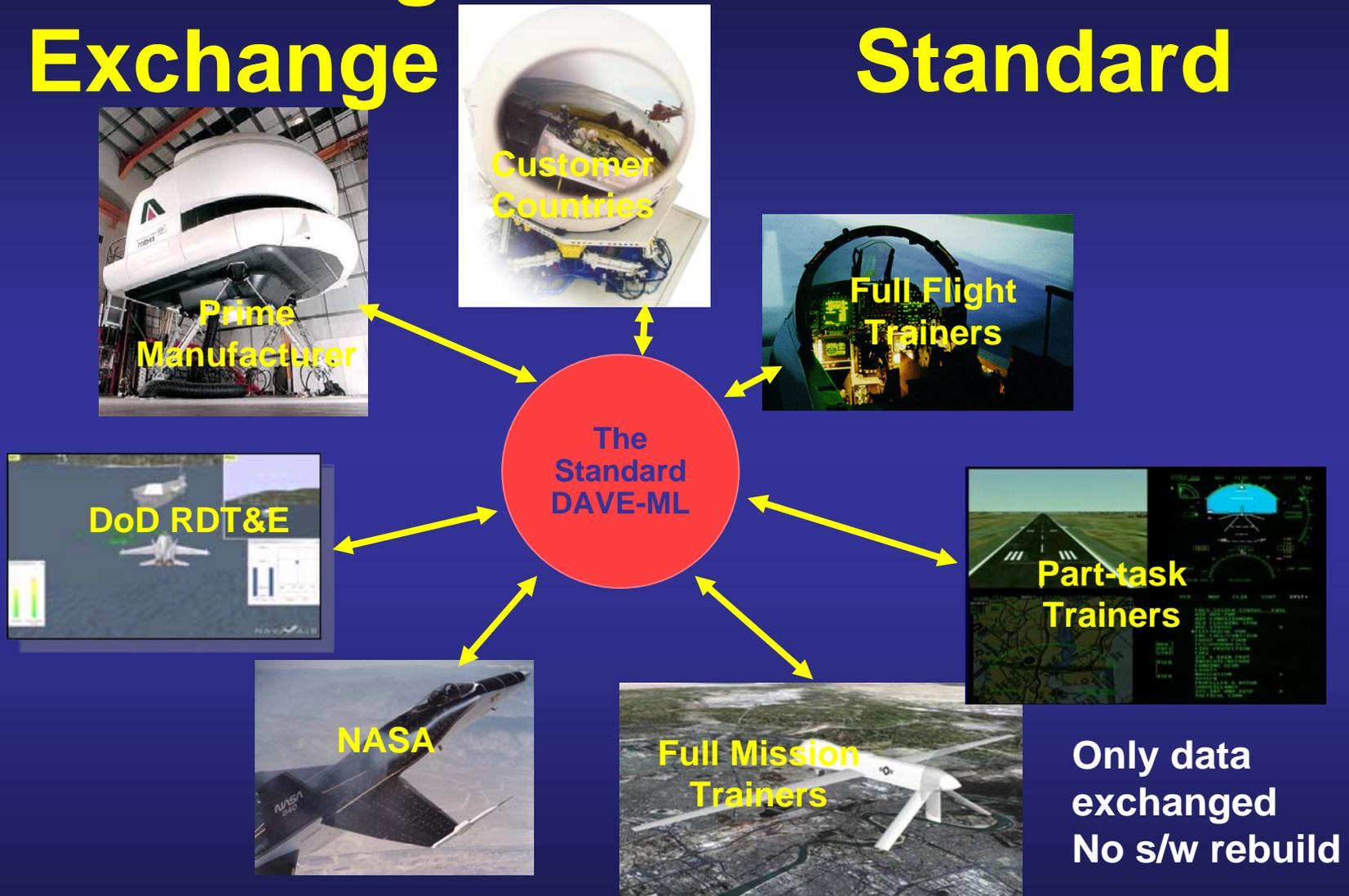
Collaboration is sharing this information- the standard makes this easier

Understanding the Problem-how do we collaborate?



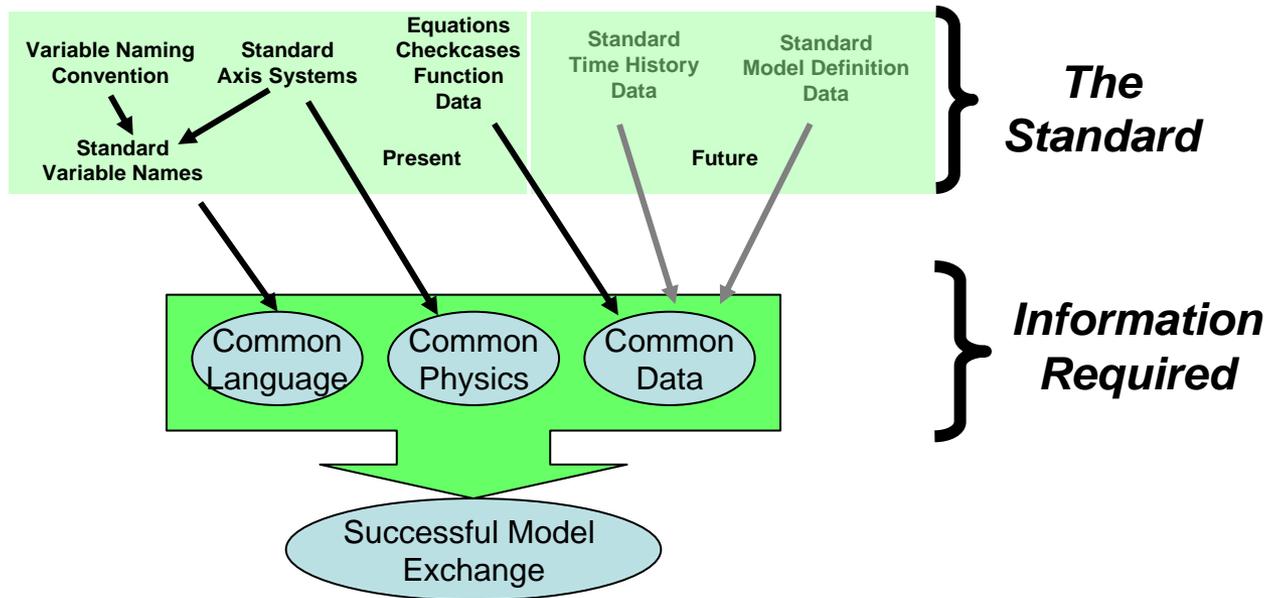
Each simulator has many different communication links

Solving the Problem- an Exchange Standard



Now Each Simulator has one fixed communication link

Information Required



More than data is required, you must define the data

Exchanging a Model Requires More than Data

- **The Standard Includes**
 - **Data**
 - **Function tables and static data**
 - **Function table check cases**
 - **Simple model definition (simple equations)**
 - **Definition of the data through:**
 - **Standard variable names and detailed definition**
 - **Standard axis systems to provide the basis of definition of the variable names**

This is captured in an XML Schema named DAVE-ML

The Solution (at Least the Beginning of the Solution)

- **AIAA/ANSI S-119-200x, “Flight Dynamics Model Exchange Standard”**
- **Status**
 - **Sponsored by the American Institute of Aeronautics and Astronautics (AIAA) Modeling and Simulation Technical Committee**
 - **Has completed public review**
 - **In final editing and production**

Standard Exchange Format

- W3C/ISO standards-based
 - XML markup grammar (DAVE-ML)
 - Reuses MathML 2 for equations
 - ANSI/AIAA Recommended Practice R-004-1992 (Axis Systems)
 - ISO 1151-1 through 9:1982-1998 (Concepts and quantities)
 - ISO 31-0:1992-1998 (Quantities and units)
 - ISO 8601:2004 (date formats)
- Implements other elements of the Standard

A plethora of XML editing/parsing toolsets are available



Features of DAVE-ML

- XML grammar (defined in a DTD)
- *n*-dimensional function tables
 - both gridded and ungridded
- Buildup equations
- Extensible function definitions
- Provenance and data references
- Modification log/record
- Verification/checkcase data

A DAVE-ML model is a...

- Human readable
- Machine readable
- Standalone
- Archivable
- Software-language agnostic
- Simulation-hardware agnostic
- Simulation-framework agnostic

subsystem model definition text file

Typical DAVE-ML application

- Most likely DAVE-ML application is an aero model for engineering or training simulation
- Also can be used in analysis tools; Simulink import tool available
- Can model other static subsystems (e.g. mass/inertia, engine thrust table)
- *Possible* to model dynamic elements, too (serves as nonlinear function)

Three Major Components

- **Standard axis systems**
- **Standard variable naming convention and names**
- **Standard functions and static equation implementation**

Standard Axis Systems

Earth centered Inertial (Geocentric inertial)	Ei
Earth centered earth fixed (Geocentric Earth fixed)	Ge
Vehicle-carried Orbit defined	Vo
Vehicle-carried normal Earth	Ve
Body	Body
Wind	Wind
Stability	Sa
Flight Path	Fp
total Angle of Attack	Aa
Locally Level	LI
Flat Earth	Fe
Structural	St

Clearly defined and used in the variable names

Standard Variable Naming Convention

- Methodology for creating variable names
- 9 Components if all possibilities of naming are used
- Table of already defined variables

Object is to simplify and clarify communication

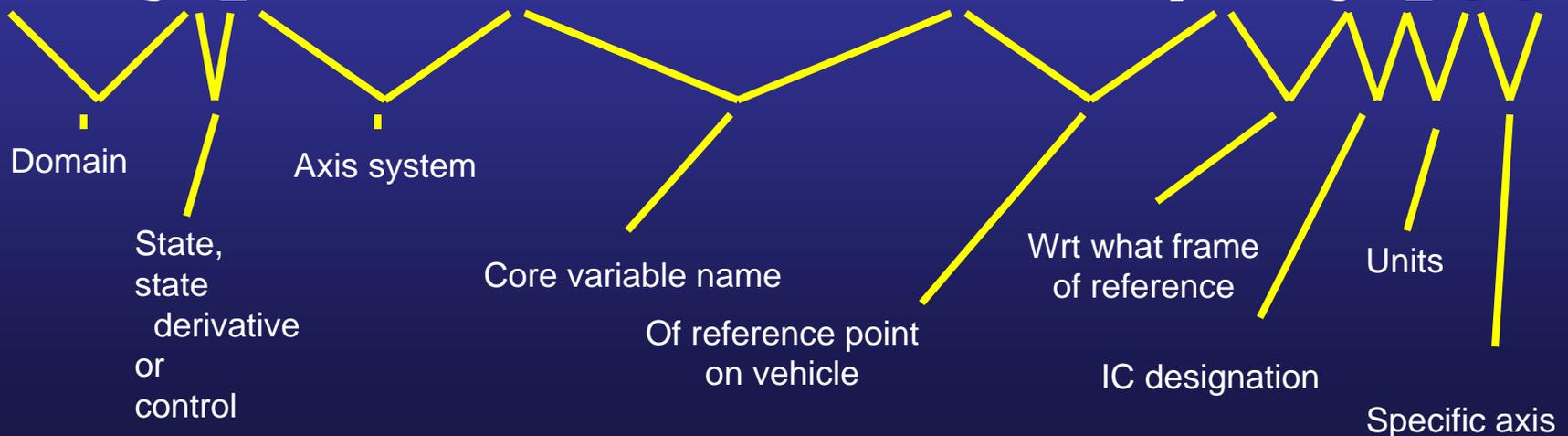
Variable Name Examples

- Common Examples

- `angleOfAttack_r`
- `bodyAccel_d_s2_Pitch`
- `s_BodyAccel_r_s2[Roll]` or `s_BodyAccel_r_s2[0]`

- Full name examples- Maximum of 9 components

- `config_s_StructuralReferencePositionOfPilotEyeWrtCgIc_f_X`
- `config.s_StructuralReferencePositionOfPilotEyeWrtCgIc_f[X]`
- `config.s_StructuralReferencePositionOfPilotEyeWrtCgIc_f[0]`



Benefits-Standard Variable Names

- Compare

Standard variable names in yellow



```
CLFlapsUp = CLALFA*angleOfAttack + CLDe *  
* de + CLQ*pRate*Chord/(2.0>trueAirspeed)
```

Vs

```
CLFlapsUp = CLALFA__r*angleOfAttack_r + CLDe__d *  
de_d + CLQ*pRate_r_s*Chord_f  
/(2.0>trueAirspeed_f_s)
```

Vs

```
CLFlapsUp = CLALFA__r*angleOfAttack_r + CLDe__d *  
de_d + CLQ *pRate_r_s *Chord_f/  
(2.0>trueAirspeed_m_s)
```

Vs

```
CLFlapsUp = CLALFA_r_1*angleOfAttack_r + CLDe__d *  
averageElevatorDeflection_d +  
CLQ*bodyRate_r_s_Pitch*referenceWingChord_m  
/(2.0>trueAirspeed_m_s)
```

The use of units and standard variable names makes this code much clearer, therefore communication is clearer

Benefits-Standard Variable Names

- Example: which is clearer?

- fuelTankLocation [4,3] =

- fuelTankCentroidWrtMrc_m[4,3] =

Standard variable names in yellow

Standard Variable Name Definition

fuelTankCentroidWrtMrc_f [number of fuel tanks,3]	Matrix used to locate the centroids of the fuel tanks. Each vehicle tank is normally numbered and the matrix should be ordered according to fuel tank number. The second component is the x, y and z moment arms from the moment reference center to the tank centroid in the vehicle structural axis. In the absence of tank numbering, the convention of port to starboard, upper to lower, then front to rear should be used.	Tank centroid behind, right, and below the moment reference center
--	--	--

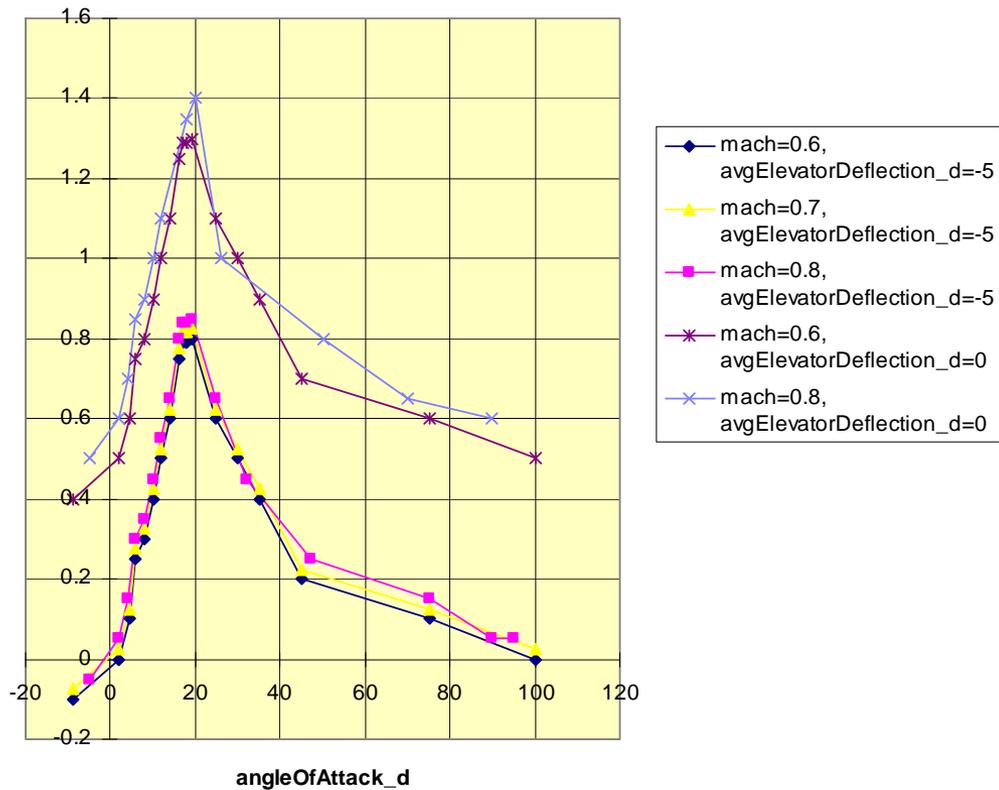
The use standard variable names makes communication easier

Flexible function tables

- **Separate definitions of function data**
- **Separate definitions of breakpoint sets**
- **Joined together to define a non-linear function**
- **Various interpolation order and extrapolation specifications**
- **Unlimited number of dimensions**
- **Can include checkcases**

Benefits-DAVE-ML Functions

CLALFA(angleOfAttack_d,mach,avgElevatorDeflection_d)

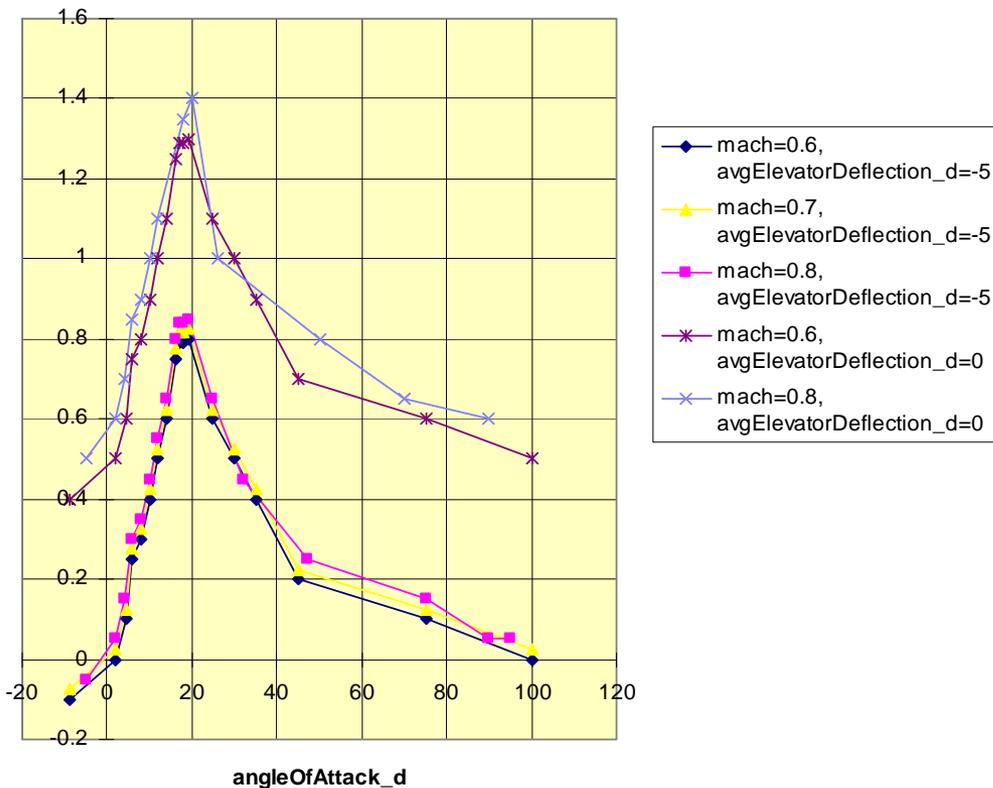


- What is missing?

The usefulness of data is very limited without the information behind the data

Benefits-DAVE-ML Functions

CLALFA(angleOfAttack_d,mach,avgElevatorDeflection_d)



- The standard adds
 - Provenance
 - References to the source data
 - Authorship
 - Change documentation
 - Confidence Intervals
 - Check cases
 - Hooks to standard variables
- Import/Export tools- a standard mechanization for access

The standard encourages inclusion of provenance and check cases

Flexible function definitions

- Uses MathML calculation grammar
- Calculations can depend on tables, or vice-versa
- Support for constants, inputs, internal calculations, and output variables
- Checkcases can include internal values
- MathML function definitions can be extended for aerospace (e.g. atan2, ...)

Typical usage

- Develop script (PERL, Java, Ruby, Python) to export from original aero data source format to DAVE-ML; mostly automated with some manual editing
- Need to generate checkcase data
- DAVE-ML then serves as new source document
- Develop (one time) script to import
- Can be shared quickly with others

KEY: Import/Export tools only need to be created once!

Examples of DAVE-ML applications

- **NASA:**
 - F-16 subsonic aero (non-linear)
 - HL-20 lifting body aero (Mach 0.5 – 4.0)
 - Blended-Wing-Body aero (750 K data points)
 - Orion capsule and launch abort stack
- **DSTO: Threat models**
- **US Navy: NextGen Threat System (NGTS)**



Available DAVE-ML tools

- **Janus**: software API to read/write/verify DAVE-ML models (courtesy DSTO)
- **DAVEtools**: Java package to evaluate DAVE-ML models; can generate Simulink models (NASA LaRC)
- **GENESIM**: can generate C++ models from DAVE-ML (SourceForge)

See complete list at daveml.nasa.gov

Available XML tools

Roll-your-own API by using

- libxml2 (open source) C++ library for XML import/export
- Jdom (open source) Java package for XML import/export

Generic XML editing packages

- oXygen XML editor (commercial; also supports Eclipse IDE plug-in)

Summary-Benefits

- **Improved fidelity through collaboration**
 - Improved model fidelity results in improved training
- **Lower cost of simulation**
 - Development- use existing models
 - Support
 - Facilitates tech refresh
 - Easy to incorporate model improvements developed elsewhere
 - Improved simulator documentation
- **Low cost/risk to using the standard**

Low risk to adopt- high potential return

The Future

- **Standard must be maintained and changed to meet user requirements**
 - More standard axis systems
 - More standard variable names
- **Time history data exchange-Standard has static function check capability- needs time history data format exchange capability to facilitate simulation validation**
- **Complete Dynamic Model Definition- something like MATLAB and Simulink**

A standard must grow to meet future requirements

Resources

- Project website: <http://daveml.nasa.gov>
- Simulation standards discussion list: sim-stds-subscribe@lists.nasa.gov or view archives at website

Coming soon: AIAA S-119 website

Backup

Benefits-Standard Variable Name Convention

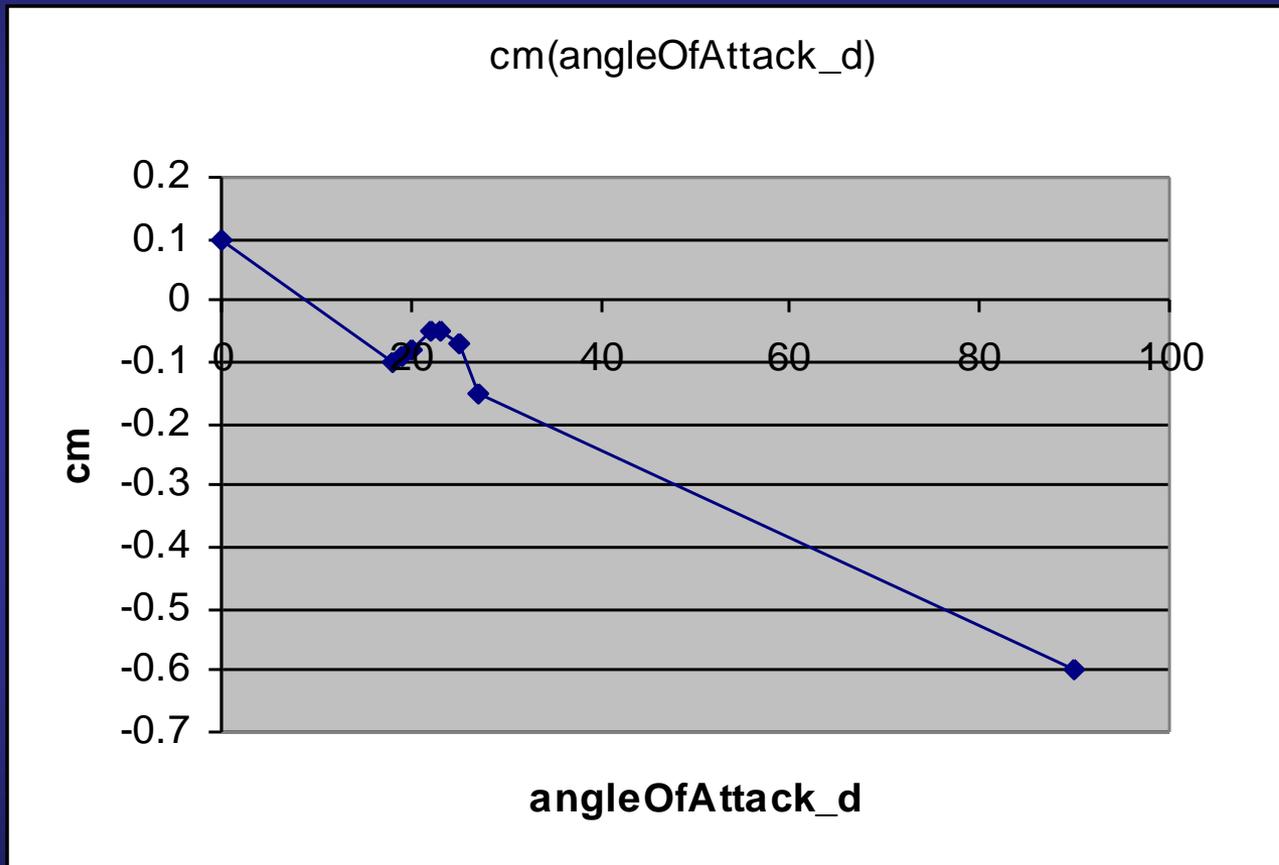
- Velocity or airspeed
- UBodyVelocity
- UBodyVelocity_f_s (std defaults to GG and inertial references)
- UBodyVelocityWRTInertial_f_s
- UBodyVelocityOfCGWRTInertial_f_s
- UBodyVelocityOfPitotProbeWRTWind_f_s
- totalVelocityOfCGWRTGround_f_s
- XGEVelocity_m_s
- Altitude
- heightOfCGWRTTerrain_f
- heightOfCGWRTMSL_f
- heightOfRadAltWRTTerrain_f
- longitudeOfIMUWRTWGS84_d
- longitudeOfCGWRTWGS84_d
- longitudeOfPilotEyeCGWRTWGS84_d

**Standard
variable
names in
yellow**

Axis system references

The use of units and standard variable names makes this code much clearer, therefore communication is clearer

Function Example



Function Example- FileHeader component

```
<fileHeader>
```

```
  <author name="Bruce Hildreth" org="SAIC"  
    email="bruce.hildreth@saic.com"/>
```

```
  <fileCreationDate date="2006-03-18"/>
```

```
  <description>
```

This is made up data to use as an example of a simple gridded function.

```
  </description>
```

```
  <reference refID="BLHRpt1" author="Joe Smith"  
    title="A Generic Aircraft Simulation Model (does not really  
    exist)"
```

```
    accession="ISBN 1-2345-678-9" date="2004-01-01"/>
```

```
  <!-- no modifications so far, so we don't need a modificationRecord  
    yet -->
```

```
</fileHeader>
```

Function Example- Variable Definitions

```
<!-- Input variable -->
```

```
<variableDef name="Angle of attack" varID="angleOfAttack_d"  
  units="deg" >  
  <isStdAIAA/>  
</variableDef>
```

```
<!-- Output (function value) -->
```

```
<variableDef name="Pitching moment coefficient due to angle of attack"  
  varID="CmAlfa" units="nondimensional" sign="+ANU">  
  <description>  
    The derivative of total pitching moment with respect to  
    angle of attack.  
  </description>  
</variableDef>
```



Function Example-Breakpoint Definition

```
<breakpointDef bpID="angleOfAttack_d_bp1">  
  <description>  
    Angle of attack breakpoint set for CmAlfa, CdAlfa, and CIAlfa  
  </description>  
  <bpVals>  
    0, 18, 19, 20, 22, 23, 25, 27, 90  
  </bpVals>  
</breakpointDef>
```



Function Example-Gridded Table Definition

```
<griddedTableDef gtID="CmAlfa_Table1">
  <description>
    The derivate of Cm wrt fuselage AOA in degrees
  </description>
  <provenance>
    <author name="Jake Smith" org="AlCorp"/>
    <functionCreationDate date="2006-12-31"/>
    <documentRef refID="BLHRpt1" />
  </provenance>
  <breakpointRefs>
    <bpRef bpID="angleOfAttack_d_bp1" />
  </breakpointRefs>
  <uncertainty effect="percentage">
    <normalPDF numSigmas="3">
      <bounds>12</bounds>
    </normalPDF>
  </uncertainty>
  <dataTable>  <!-- Always comma separated values -->
    0.1,-0.1,-0.09, -.08, -0.05, -0.05, -0.07, -0.15, -0.6
  </dataTable>
</griddedTableDef>
```

Function Example-Function Definition

- `<function name="Cm_alpha_func">`
- `<description>`
- Variation of pitching moment coefficient with angle of attack (example)
- `</description>`
- `<independentVarRef varID="angleOfAttack_d"/>`
- `<dependentVarRef varID="CmAlfa"/>`
- `<functionDefn>`
- `<griddedTableRef gtID="CmAlfa_Table1"/>`
- `</functionDefn>`
- `</function>`

Function Example-Check Cases

```
<checkData>
  <staticShot name="case 1">
    <checkInputs>
      <signal>
        <varID>angleOfAttack_d</varID>
        <signalValue> 5.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
        <varID>CmAlfa</varID>
        <signalValue>0.04444444</signalValue>
        <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 2">
    <checkInputs>
```

