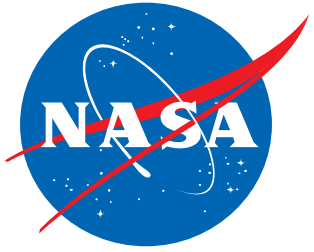


Financial Impact of ANSI/ AIAA-S-119-2011 Flight Dynamic Model Exchange Standard

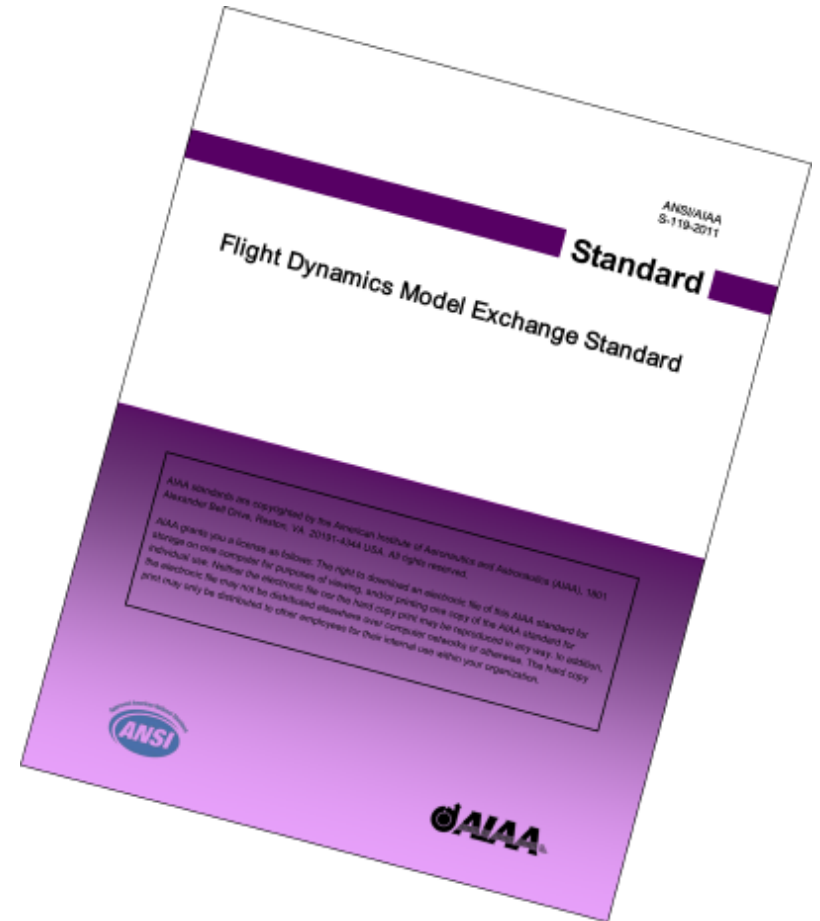
Bruce Jackson, NASA Langley
<bruce.jackson@nasa.gov>

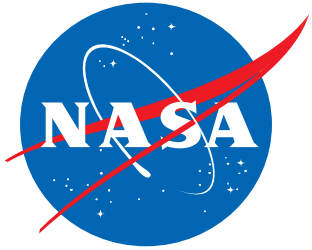
October 2011



Outline

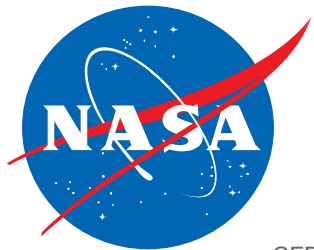
- Motivation
- Purpose
- Background
- Solution
- Features
- Status
- Tools
- Implementation case study
- Estimated DoD savings opportunity



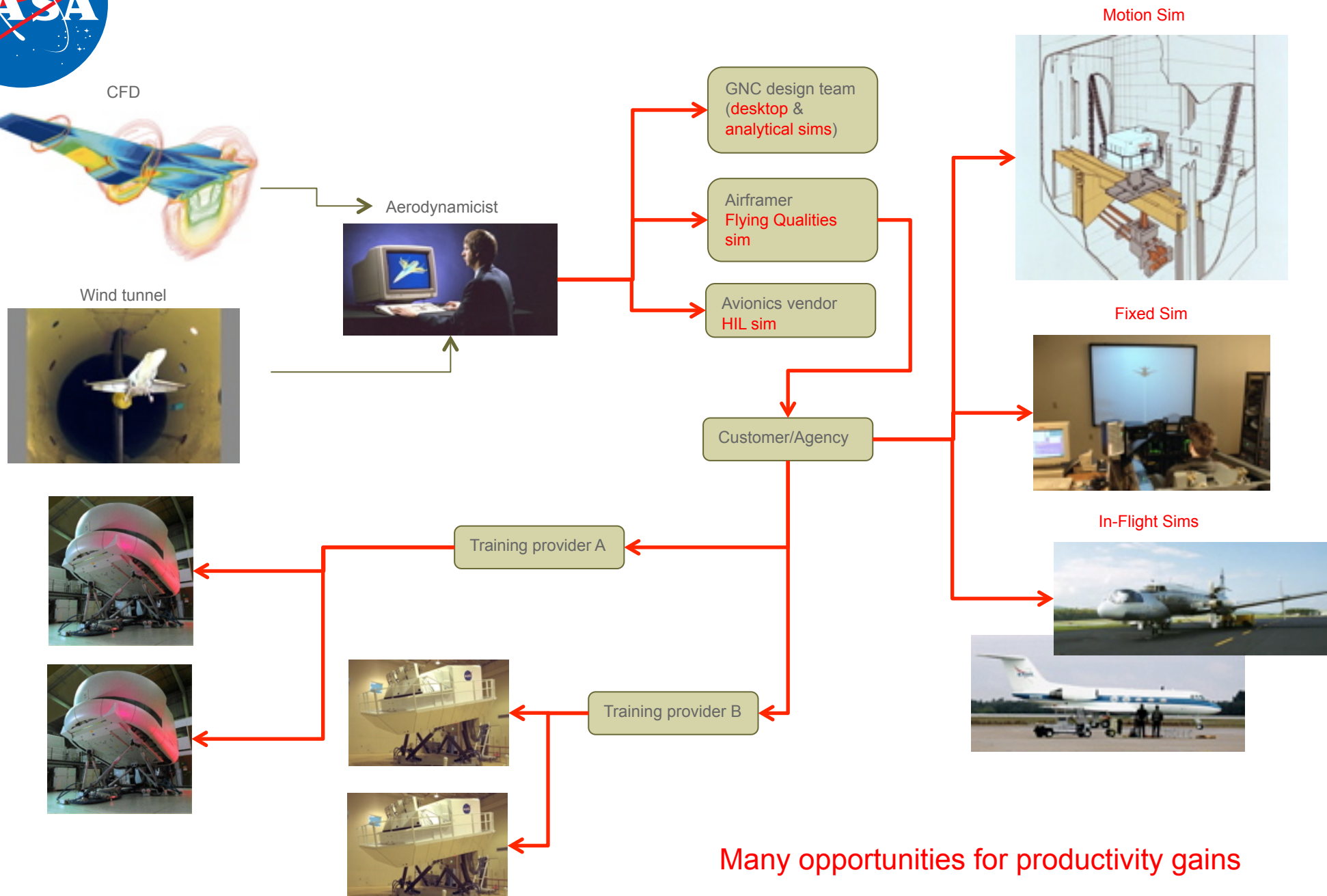


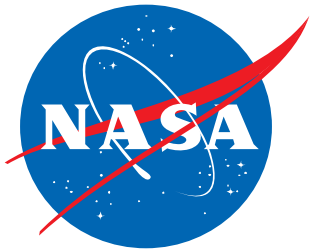
Motivation

- Growth of teaming within aero industry
 - Exponential increase in desktop compute power
 - Increasing reliance on sim-based acquisition
 - Emergence of web-based everything
-
- 1980's simulation technology hasn't kept pace



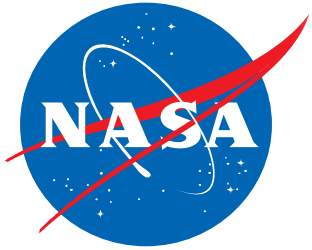
Model data flow (notional)





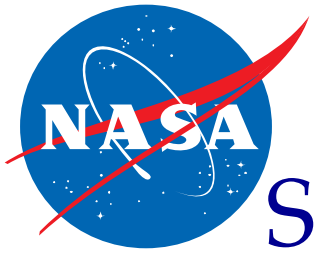
Purpose

- Agree on a format for the **exchange** of aerospace simulation databases & models, using standard UNICODE strings for:
 - axis systems: wind, sa, ...
 - units of measure: deg, lbf_ft2, ...
 - common variable names: angleOfAttack_deg
- Focus on low-hanging fruit: aerodynamic models
- Also works for mass/inertia models, engine thrust tables; basically any static non-linear function
- Uses custom XML grammar, **DAVE-ML** = **D**ynamic **A**erospace **V**ehicle **M**arkup **L**anguage



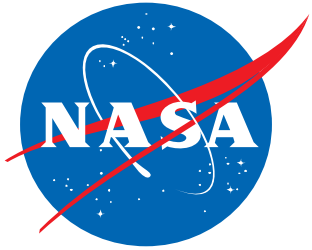
Background

- Bruce Hildreth & Bruce Jackson stood up Navy's Manned Flight Simulator (Pax River) c. 1982
- Imported lots of fighter, other models (F-4, F-14, X-29, AV-8B, F/A-18A, V-22, A-6F)
- Each data package used unique variable names, axis conventions, data formats, coding style
- Each took **6-8 months** to rehost to Pax's CASTLE framework – **there's gotta be a better way!**
- Hildreth's 1998 AIAA paper spelled out requirements
- 2004 AIAA paper showed \$6 M savings opportunity *per aircraft type*



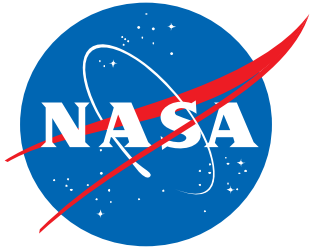
Solution: DAVE-ML (XML grammar)

- Open standard (W3C)
- Transportable to any platform
- Human- and machine-readable UNICODE
- Programming language agnostic
- Same source used for detail design & pilot training
- Good fit for databases and models
- Archive-able history of model (via metadata)
- Transformable (->HTML, etc.) via XSLT
- Lots of generic XML tools, editors, parsers, validators



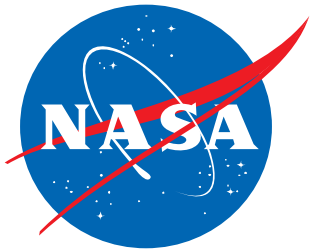
S-119 / DAVE-ML features

- Non-linear function descriptions
 - Tabular (traditional aero models)
 - » Unlimited dimensionality
 - » Gridded or unstructured
 - Polynomial or exponential expressions
- Model build-up equations (using MathML)
- Units of measure
- Both 'standard' and internal variable names for same variable
- Uncertainty metrics included
- Provenance (history / source of data)
- Checkcase / verification data included



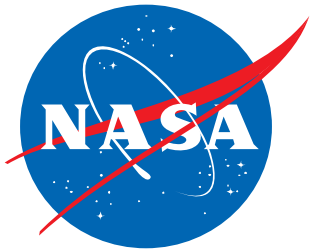
Status

- Approved by AIAA February 2011
- Approved by ANSI March 2011
- De-facto NASA standard (JSC, MSFC, LaRC, DFRC, GFRC sim tools / frameworks are compatible) after 2010 NESC assessment
- Australia's DSTO adopted S-119 exclusively
- Some tools available, more needed
- More improvements planned



Tools for using S-119

- *JANUS* API (C++ API library)
Australian DSTO/Ball Aerospace
- NASA Ames, Johnson, Dryden real-time sims
- NASA Langley *DaveMLTranslator* C++ API
- NASA Marshall *MAVERIC* sim
- NASA Glenn *OTIS4* trajectory optimization tool
- NAVAIR's *CASTLE* sim shell (latest release)
- XSLT conversion scripts
- DAVEtools (Java package)
- Several example aero models (F-16, HL-20, others)
- Visit www.daveml.org for more info



Existing S-119 examples



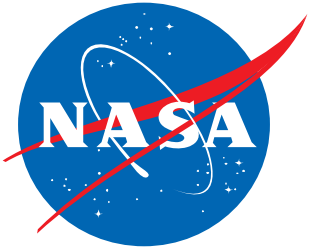
F-16 subsonic aero model

- 51 variables, 18 tables, 744 points
- Switches & absolute-value nonlinear elements
- 17 verification check-cases included
- 154 KB file with 2,712 lines

HL-20 lifting body aero model

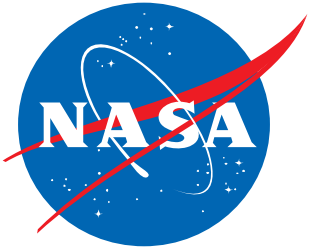
- Supersonic and subsonic regimes
- 361 variables, 168 tables, 6,240 points
- 25 verification check-cases included
- 1.2 MB file with 22,299 lines





Other models in S-119

- Blended-Wing-Body aero model (proprietary)
- Ares I booster and capsule aero model (ITAR)
- Twin-engine commercial jet transport aero, inertia models (full envelope)
- Reduced-fidelity aero models of:
 - A-7
 - F-4
 - F-14
 - F-106
 - X-31



XSLT conversion scripts

- eXtensible Stylesheet Language Transformation
- XSLT scripts, themselves XML documents, specify how to convert XML-based files into other formats.
- DAVE `html.xml` provides a conversion from DAVE-ML XML file into an XHTML document
- `DMLtoC.xml` does conversions directly to C++
- Requires `xerces` or similar XSLT tool
- Scripts available for download =>
<http://www.daveml.org>



Janus

Janus was the god of [gates](#), [doors](#), [doorways](#), [beginnings](#), and [endings](#).

(from Wikipedia)



- C++ interface class for DAVE-ML datasets
- Enables DAVE-ML as native format – not just exchange
- Abstracts data from model code – component to total effect
- Common API for dissimilar data structures
- Centralises data handling functions
- Implements Gridded Data, Ungridded Data and MathML



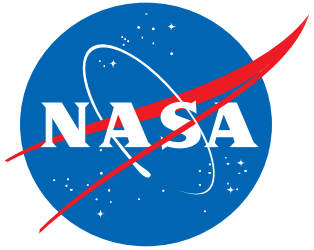
Janus

- Validates dataset against DTD, then parses & loads into a Document Object Model (using Apache *Xerces* parser)
- Build numerical structures in memory for variables / tables
- Manages inputs, outputs and internal variables
- Performs specified interpolation / extrapolation
 - Discrete, Linear, Quadratic, Cubic
 - Add Spline and other basis functions in the future.



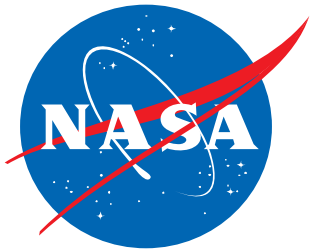
Janus

- Encryption/decryption capability using AES-256bit symmetric keys and RSA algorithm.
 - Impacts on instantiation but NO runtime penalty.
 - Used to ensure data integrity.
- Performance for >> real-time applications.



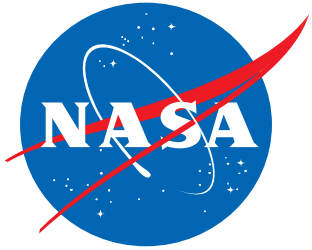
DAVEtools 0.9

- Single Java `DAVEtools.jar` archive file
 - Base package, **DAVE**
 - » Builds internal objects from DAVE-ML description
 - » Performs auto-verification of model
 - » Allows simple exercising of model
 - » Callable from Matlab® to operate model
 - Simulink® export tool, **DAVE2SL**
 - » Converts DAVE-ML models into Simulink® blocks
 - Response surface viz tool, **ModelSweeper**
- Java 1.6 source code available from NASA under NASA Open Source license



Implementation Case Study

- As part of 2010 NASA assessment, five R&D simulation facilities were asked to **add S-119 model import capability**
- Estimated cost @ 0.5 staff-year per site (~2.5 total)
- Actual cost per site ranged from 1 week to 3 months
- Underspent funding by 65%
- Continued, ongoing savings for future programs, since now nearly all NASA flight simulation tools and facilities can import this format

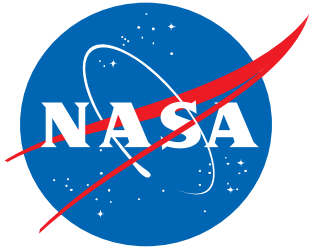


Cost savings estimate

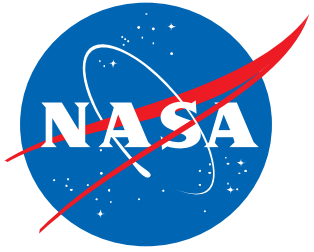
- In 2002 there were 59 different F / A-18 simulations (training / real-time, R&D / analytical)
- Little commonality in physics model except original source
- 2004 estimate showed \$6 million per year savings opportunity from common sim model format¹
- Savings estimated from
 - Reduced negative training / incorrect analyses
 - Improved productivity
 - Aero database maintenance improvements (faster distribution, model updates easier)

\$ 6,000,000 per year savings for one type of aircraft

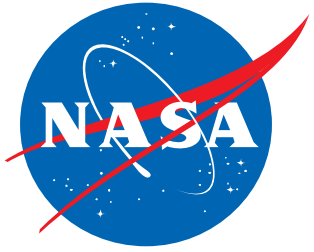
Ref 1: AIAA 2004-5038, Jackson, Hildreth, York, Cleveland:
*Evaluation of a Candidate Flight Dynamics Model Simulation
Standard Exchange Format*



Backup Slides

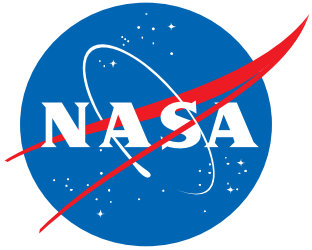


DAVE-ML OVERVIEW



DAVE-ML grammar

- Top-level syntax shown
- Specified by *DAVEfunc.dtd* Document Type Definition
- Examples taken from reference manual, available for download from <http://www.daveml.org> - see "DTDs"



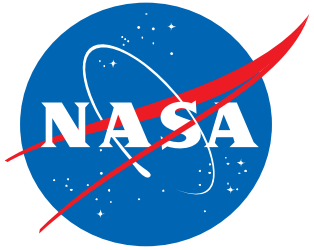
Top-level syntax

- Top level-element is <DAVEfunc>

```
DAVEfunc :  
    fileHeader :  
    variableDef+ :  
    breakpointDef* :  
    griddedTableDef* :  
    ungriddedTableDef* :  
    function* :  
    checkData? :
```

Key: *element : attribute [optional attribute]*
 subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*



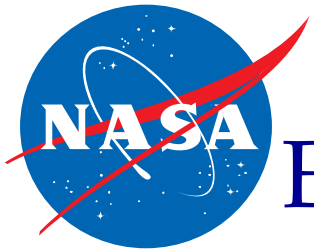
File header element

- Provides documentation trail for model

```
fileHeader : [name]
             author+ : name, org
             creationDate : date
             fileVersion? :
             description? :
             reference* :
             modificationRecord* :
             provenance* :
```

*Key: element : attribute [optional attribute]
subelement*

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*



Example of fileHeader element

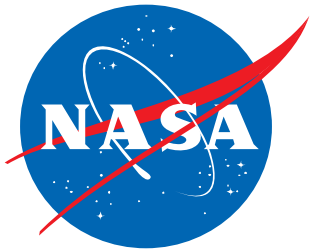
```
<!--=====-->
<!--===== FILE HEADER =====-->
<!--=====-->
```

```
<fileHeader>
  <author name="Bruce Jackson" org="NASA Langley Research Center"
    email="bruce.jackson@nasa.gov">
    <address>MS 132 NASA, Hampton, VA 23681</address>
  </author>
  <creationDate date="2003-03-18"/>

  <fileVersion>$Revision: 233 $</fileVersion>

  <description>
    Version 2.0 aero model for HL-20 lifting body, as described in
    TM-107580. This aero model was used for HL-20 approach and
    landing studies at NASA Langley Research Center during 1989-1995
    and for a follow-on study at NASA Johnson Space Center in 1994
    and NASA Ames Research Center in 2001. This DAVE-ML version
    created 2003 by Bruce Jackson to demonstrate DAVE-ML.
  </description>

  <reference refID="REF01"
    author="Jackson, E. Bruce; Cruz, Christopher I. & and Ragsdale, W. A."
    title="Real-Time Simulation Model of the HL-20 Lifting Body"
    accession="NASA TM-107580"
    date="1992-07-01"
  />
```

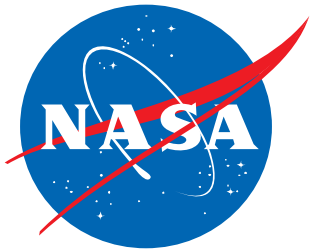


fileHeader example, cont'd

```
<reference refID="REF02"
  author="Cleveland, William B. <nospam@mail.arc.nasa.gov>"
  title="Possible Typo in HL20_aero.xml"
  accession="email"
  date="2003-08-19"
/>
.
.
.

<modificationRecord modID="A" refID="REF02" date="2002">
  <author name="Bruce Jackson" org="NASA Langley Research Center"
    email="bruce.jackson@nasa.gov">
    <address>MS 132 NASA, Hampton, VA 23681</address>
  </author>
  <description>
    Revision 1.24: Fixed typo in CLRUD0 function description which
    gave dependent signal name as "CLRUD1." Bill Cleveland of NASA
    Ames caught this in his xml2ftp script. Also made use of 1.5b2
    fileHeader fields and changed date formats to comply with
    convention.
  </description>
</modificationRecord>
.
.
.

</fileHeader>
```



Variable Definition element (variableDef)

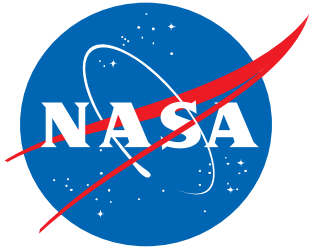
- Used to define each variable used in the model
 - Includes inputs, outputs, constants, parameters and local (temporary) variables

```
variableDef : name, varID, units,  
              [axisSystem, sign, alias, symbol, initialValue]  
description? : (description character data)  
(provenance | provenanceRef)? : (historical information)  
calculation? : math (defined in MathML2.0 DTD)  
(isInput | isControl | isDisturbance)? :  
isOutput? :  
isState? :  
isStateDeriv? :  
isStdAIAA? :  
uncertainty? : effect (additive | multiplicative | percentage | absolute)  
                  (normalPDF : numSigmas | uniformPDF : symmetric )
```

Key: *element : attribute [optional attribute]*

subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*

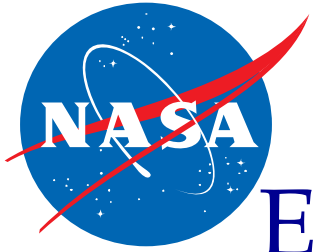


Example of variableDef for two input elements

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

<variableDef name="mach" varID="XMACH" units="nd" symbol="M">
  <description>
    Mach number (dimensionless)
  </description>
  <isStdAIAA/>
</variableDef>

<variableDef name="lowerLeftBodyFlapDeflection"
  varID="DBFLL" units="deg" sign="ted" symbol="θbfl1">
  <description>
    Lower left body flap deflection, deg, +TED (so deflections are
    always zero or positive).
  </description>
</variableDef>
```

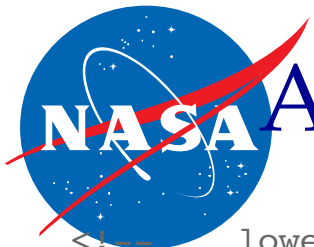


Example of local variable definition

```
<!-- PRELIMINARY BUILDUP EQUATIONS -->

<!-- LOWER LEFT BODY FLAP CONTRIBUTIONS -->

<!--      table output signal      -->
<variableDef name="Cldbfl1_0" varID="CRBFLL0" units="nd">
  <description>
    Output of CRBFLL0 function; rolling moment contribution of
    lower left body flap deflection due to  $\alpha^0$  (constant
    term).
  </description>
</variableDef>
```



A more complex variable definition

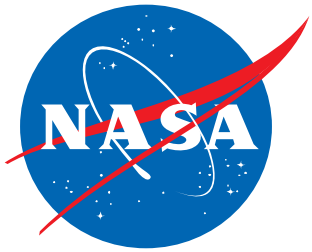
```

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

```

This MathML2 encodes:

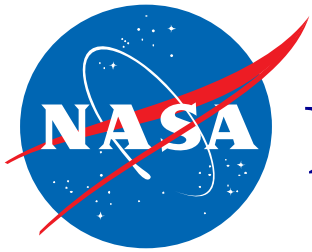
$$C_{L\partial BFLL} = C_{L\partial BFLL0} + \alpha C_{L\partial BFLL1} + \alpha^2 C_{L\partial BFLL2} + \alpha^3 C_{L\partial BFLL3}$$



An output variable definition

```
<!-- ===== -->  
<!-- Output variables -->  
<!-- ===== -->
```

```
<variableDef name="totalCoefficientOfLift" varID="CL" units="nd" sign="up" symbol="CL">  
  <description>  
    Coefficient of lift  
    CL = CL0 + CLBFUL + CLBFUR + CLBFL + CLBFLR +  
          CLWFL + CLWFR + CLRUD + CLGE + CLLG  
  </description>  
  <calculation>  
    <math>  
      <apply>  
        <plus/>  
        <ci>CL0</ci>  
        <ci>CLBFUL</ci>  
        <ci>CLBFUR</ci>  
        <ci>CLBFL</ci>  
        <ci>CLBFLR</ci>  
        <ci>CLWFL</ci>  
        <ci>CLWFR</ci>  
        <ci>CLRUD</ci>  
        <ci>CLGE</ci>  
        <ci>CLLG</ci>  
      </apply>  
    </math>  
  </calculation>  
  <isOutput/>  
</variableDef>
```



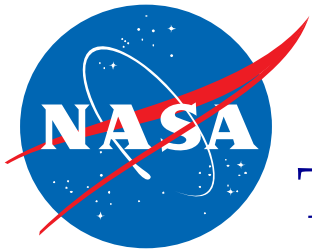
Breakpoint set definition element (breakpointDef)

- Used to define a common set of independent variable values associated with one or more tables of function data

```
breakpointDef : bpID, [name, units]
               description? : (description character data)
               bpVals : (list of comma-separated set of breakpoint values)
```

Key: *element : attribute [optional attribute]*
 subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*

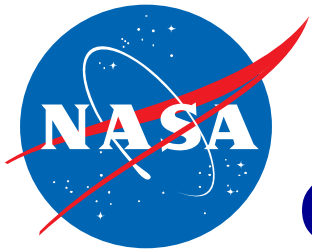


Two examples of breakpointDef elements

```
<!-- ===== -->
<!-- ===== BREAKPOINT SETS ===== -->
<!-- ===== -->
```

```
<breakpointDef name="Mach" bpID="XMACH1_PTS" units="nd">
  <description>
    Mach number breakpoints for all aero data tables
  </description>
  <bpVals>
    0.3, 0.6, 0.8, 0.9, 0.95, 1.1, 1.2, 1.6, 2.0, 2.5, 3.0, 3.5, 4.0
  </bpVals>
</breakpointDef>

<breakpointDef name="Lower body flap" bpID="DBFL_PTS" units="deg">
  <description>Lower body flap deflections breakpoints for tables</description>
  <bpVals>0., 15., 30., 45., 60.</bpVals>
</breakpointDef>
```



Gridded function table definition

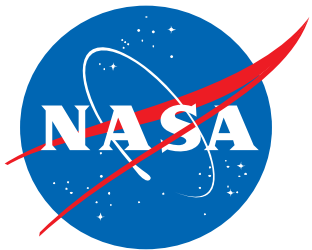
- Defines nonlinear function dependent values

```
griddedTableDef : [gtID, name, units]
  description? : (description character data)
  provenance? : [provID] (or provenanceRef if duplicating previous provenance)
    author+ : name, org
      (address* | contactInfo*) : (contact information)
    creationDate
    documentRef* : refID
    modificationRef* : modID
  description?
  breakpointRefs :
    bpRef+ : bpID
  uncertainty? : effect (additive | multiplicative | percentage | absolute)
    (normalPDF : numSigmas | uniformPDF : symmetric )
  dataTable : (list of comma-separated set of table values)
```

Key: *element : attribute [optional attribute]*

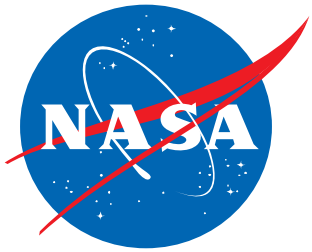
subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*



Example of griddedTableDef

```
<griddedTableDef name="CLBFL0" gtID="CLBFL0_table">
  <description>
    Lower body flap contribution to lift coefficient, polynomial constant term
  </description>
  <provenance>
    <author name="Bruce Jackson" org="NASA Langley Research Center"/>
    <creationDate date="2003-01-31"/>
    <documentRef refID="REF01"/>
  </provenance>
  <breakpointRefs>
    <bpRef bpID="DBFL_PTS"/>
    <bpRef bpID="XMACH1_PTS"/>
  </breakpointRefs>
  <dataTable> <!-- last breakpoint changes most rapidly -->
    <!-- CLBFL0      POINTS      -->
    <!-- DBFL =      0.0          -->
    0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
    0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
    0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
    <!-- DBFL =      15.0         -->
    -0.86429E-02 , -0.10256E-01 , -0.11189E-01 , -0.12121E-01 , -0.13520E-01 ,
    -0.86299E-02 , -0.53679E-02 , 0.76757E-02 , 0.11300E-01 , 0.62992E-02 ,
    0.51902E-02 , 0.38813E-02 , 0.37366E-02 ,
    .
    .
    .
  </dataTable>
</griddedTableDef>
```



An aside about table ordering...

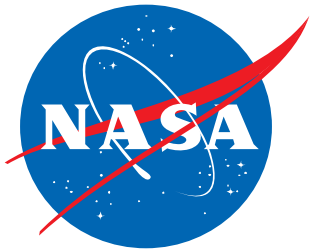
- The appearance of tables in DAVE-ML files can be confusing, in that they appear to be 2-dimensional
- In actuality, the values are a vector of the 'unraveled' n-dimensional tables with last index changing fastest
- Example: given $C_L = C_L(\alpha, M, \delta f)$, value sequence is

<datatable>

$C_L(\alpha_1, M_1, \delta f_1), C_L(\alpha_1, M_1, \delta f_2), \dots C_L(\alpha_1, M_1, \delta f_n),$
 $C_L(\alpha_1, M_2, \delta f_1), C_L(\alpha_1, M_2, \delta f_2), \dots C_L(\alpha_1, M_2, \delta f_n),$
 $C_L(\alpha_1, M_m, \delta f_1), C_L(\alpha_1, M_m, \delta f_2), \dots C_L(\alpha_1, M_m, \delta f_n),$
 $C_L(\alpha_2, M_1, \delta f_1), \dots$

</datatable>

- Line breaks can occur anywhere - they are ignored



Aside - table ordering (cont'd)

- Previous example could also be encoded

<datatable>

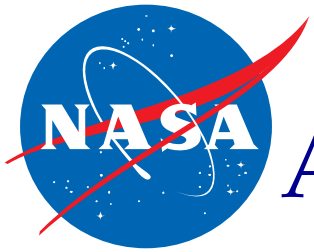
$C_L(\alpha_1, M_1, \delta f_1), C_L(\alpha_1, M_1, \delta f_2), \dots$

$C_L(\alpha_1, M_1, \delta f_n), C_L(\alpha_1, M_2, \delta f_1),$

$C_L(\alpha_1, M_2, \delta f_2), \dots C_L(\alpha_1, M_2, \delta f_n),$
 $C_L(\alpha_1, M_m, \delta f_1),$

$C_L(\alpha_1, M_m, \delta f_2), \dots C_L(\alpha_1, M_m, \delta f_n),$
 $C_L(\alpha_2, M_1, \delta f_1), \dots$

</datatable>



Aside - table ordering (concluded)

- Order of `<bpRefs>` in `<breakpointRefs>` element **is important!**
- In previous example, order would have to be

```
<breakpointRefs>
```

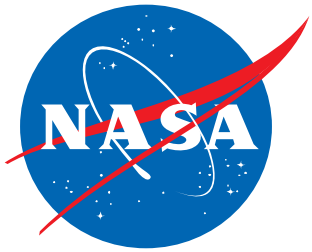
```
  <bpRef bpID="ALPHA_PTS" />
```

```
  <bpRef bpID="MACH_PTS" />
```

```
  <bpRef bpID="DELTA_FLAP_PTS" />
```

```
</breakpointRefs>
```

- ...so interpreter will know which breakpoint set to associate with which dimension of this function table.
- This is one of the few places in DAVE-ML where order is important.



Function definition element

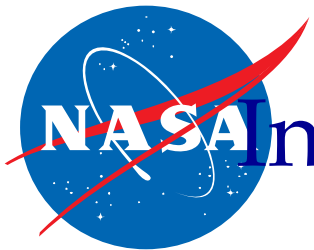
- This element ties breakpoint sets to tables
- Several different syntaxes; below is most common

```
function : name
  description? : (description character data)
  (provenance | provenanceRef)?
  independentVarRef+ : varID, [min, max, extrapolate,
    interpolate=(discrete|floor|ceiling|linear|
      quadraticSpline|cubicSpline)]
  dependentVarRef : varID
  functionDefn : [name]
  griddedTableRef : gtID
```

Key: element : attribute [optional attribute]

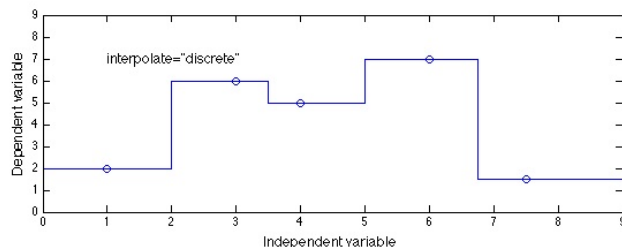
subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*

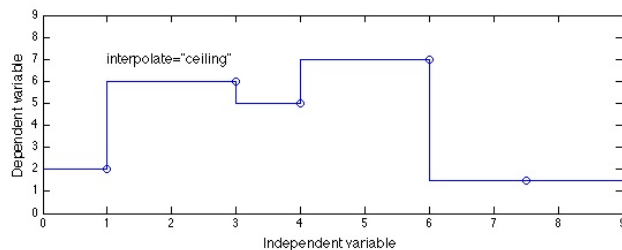


Interpolation/extrapolation options

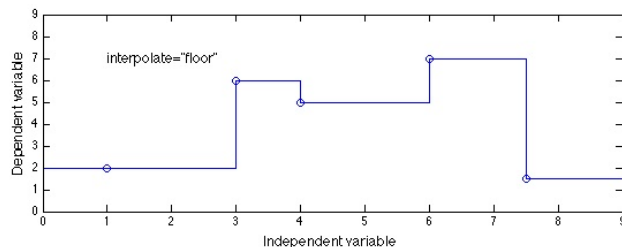
interpolate=
“discrete”



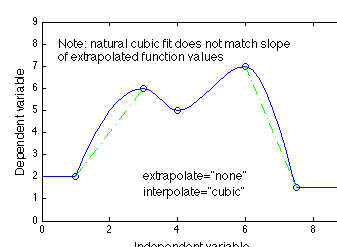
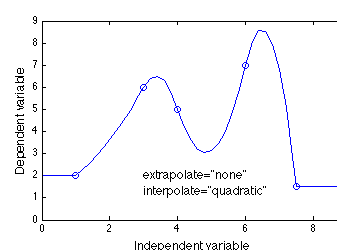
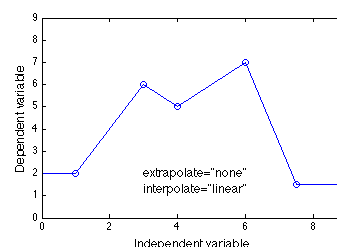
interpolate=
“ceiling”



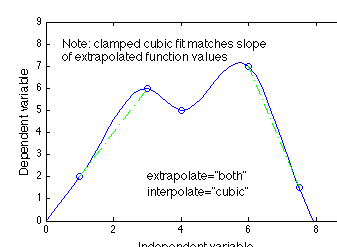
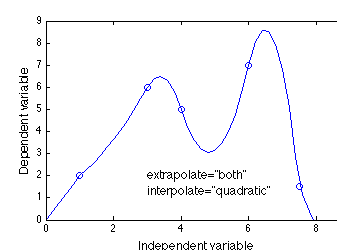
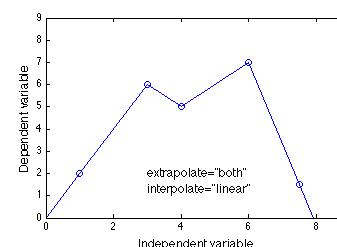
interpolate=
“floor”



extrapolate="none"



extrapolate="both"

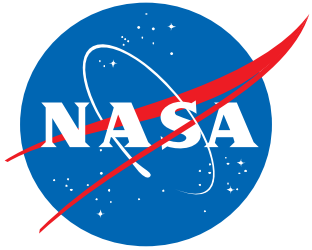


interpolate=
“linear”

interpolate=
“quadratic”

interpolate=
“cubic”

Defaults are “linear” interpolation and “none” for extrapolation



Example of function element

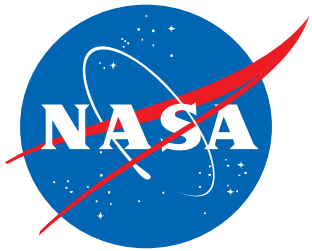
```
<!-- ===== -->
<!-- Lower left body flap functions -->
<!-- ===== -->

<function name="CLBFLL0">
  <description>
    Lower left body flap lookup function for lift, polynomial constant term.
  </description>
  <independentVarRef varID="DBFLL" min="0.0" max="60." extrapolate="neither"/>
  <independentVarRef varID="XMACH" min="0.3" max="4.0" extrapolate="neither"/>
  <dependentVarRef varID="CLBFLL0"/>
  <functionDefn name="CLBFLO_fn">
    <griddedTableRef gtID="CLBFLO_table"/>
  </functionDefn>
</function>
```

This defines a function CLBFLO_fn to be

CLBFLO_fn = CLBFLO_fn(DBFLL, XMACH)

with limits on input values and no extrapolation



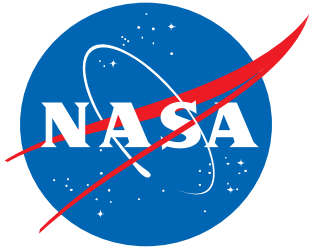
Verification data (checkData) element

```
checkData :  
  staticShot* : name, [refID]  
    checkInputs :  
      signal+ :  
        signalName :  
        signalUnits :  
        signalValue :  
      internalValues? :  
        signal+ :  
          varID :  
          signalValue :  
      checkOutputs :  
        signal+ :  
          signalName :  
          signalUnits :  
          signalValue :  
        tol :
```

Key: *element : attribute [optional attribute]*

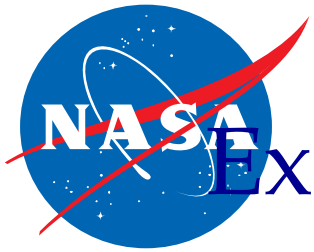
subelement

'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*



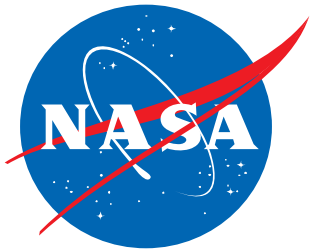
Example checkData element

```
<checkData>
  <staticShot name="Nominal" refID="NOTE1">
    <checkInputs>
      <signal>
        <signalName>true_Airspeed</signalName>
        <signalUnits>f_s</signalUnits>
        <signalValue> 300.000</signalValue>
      </signal>
      <signal>
        <signalName>angleOfAttack</signalName>
        <signalUnits>deg</signalUnits>
        <signalValue> 5.000</signalValue>
      </signal>
      <signal>
        <signalName>bodyAngularRate_Pitch</signalName>
        <signalUnits>rad_s</signalUnits>
        <signalValue> 0.000</signalValue>
      </signal>
      <signal>
        <signalName>elevatorsDeflection</signalName>
        <signalUnits>deg</signalUnits>
        <signalValue> 0.000</signalValue>
      </signal>
      .
      .   (other input values)
      .
    </checkInputs>
  .
  .
  .
</checkData>
```



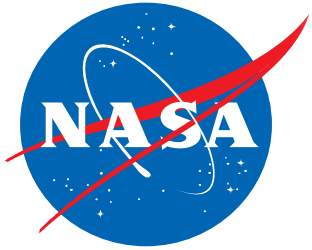
Example checkData element (cont'd)

```
.  
.   
.   
<checkOutputs>  
  <signal>  
    <signalName>aeroBodyCoefficient_X</signalName>  
    <signalValue>-0.00400000000000</signalValue>  
    <tol>0.000001</tol>  
  </signal>  
  <signal>  
    <signalName>aeroBodyCoefficient_Z</signalName>  
    <signalValue>-0.41600000000000</signalValue>  
    <tol>0.000001</tol>  
  </signal>  
  <signal>  
    <signalName>aeroBodyMomentCoefficient_Pitch</signalName>  
    <signalValue>-0.04660000000000</signalValue>  
    <tol>0.000001</tol>  
  </signal>  
  .  
  .   (other output values & tolerances)  
  .   
</checkOutputs>  
</staticShot>  
.  
.  
.  
</checkData>
```



checkData elements can have intermediate values to assist debugging

```
<checkData>
  <staticShot name="Skewed inputs">
    <checkInputs>
      .
      .   (checkcase input values)
      .
    </checkInputs>
    <internalValues>
      <signal>
        <signalID>vt</signalID>
        <signalValue>300.0</signalValue>
      </signal>
      <signal>
        <signalID>alpha</signalID>
        <signalValue>16.2</signalValue>
      </signal>
      <signal>
        <signalID>q</signalID>
        <signalValue>-0.76</signalValue>
      </signal>
      .
      .   (additional internal values; normally one for every internal variable)
      .
    </internalValues>
    <checkOutputs>
      .
      .   (checkcase output values)
      .
    </checkOutputs>
  </staticShot>
</checkData>
```



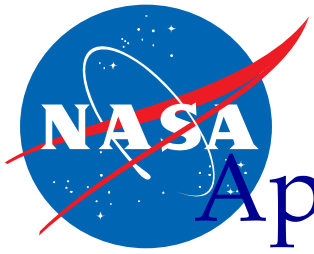
uncertainty Element

```
uncertainty : effect=['additive' | 'multiplicative' | 'percentage' | 'absolute']
  EITHER
    normalPDF : numSigmas=['1','2','3',...]
      bounds : (value corresponding to 1, 2, 3,...  $\sigma$ )
      correlatesWith* : varID
      correlation : varID, corrCoef
  OR
    uniformPDF :
      bounds : (symmetric or lower bound value)
        (dataTable | variableDef | variableRef | PCDATA)
      [bounds : (upper bound value)]
```

Key: *element : attribute [optional attribute]*

subelement

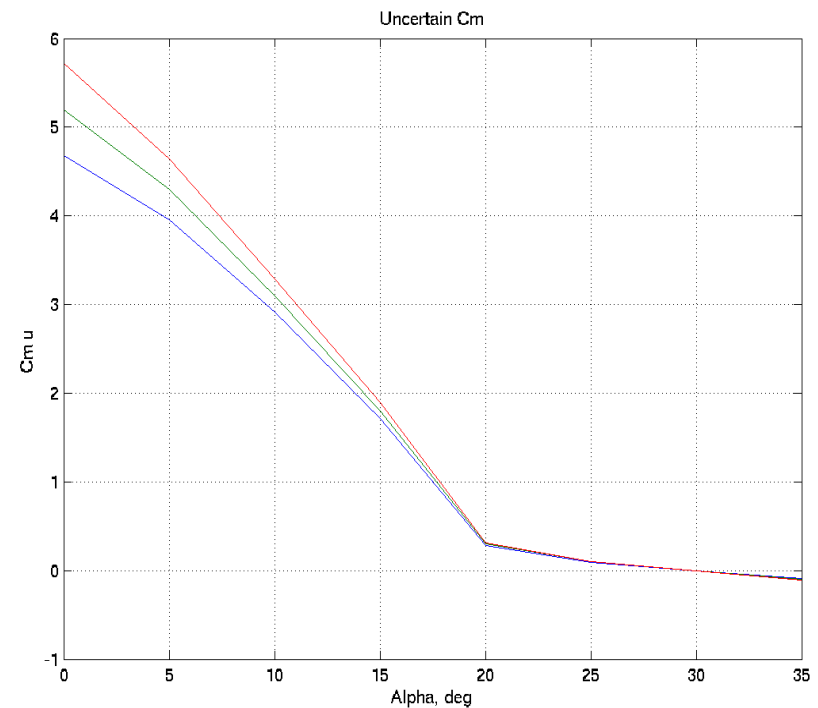
'+' means 1 or more; '' means 0 or more; '?' means 0 or 1*



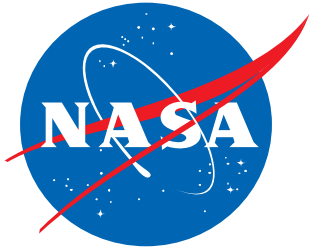
Application of uncertainty element

Example: add 5-12% uncertainty bound around C_m

```
<function name="Uncertain Cm">
  <independentVarRef varID="Alpha_deg"/>
  <dependentVarRef varID="Cm_u"/>
  <functionDefn>
    <griddedTableDef>
      <breakpointRefs>
        <bpRef bpID="ALP"/>
      </breakpointRefs>
      <uncertainty effect="percentage">
        <normalPDF numSigmas="3">
          <bounds>
            <dataTable>
              0.10, 0.08, 0.06, 0.05, 0.05,
              0.06, 0.07, 0.12
            </dataTable></bounds>
          </normalPDF>
        </uncertainty>
        <dataTable>
          5.2, 4.3, 3.1, 1.8, 0.3, 0.1, 0.0, -0.1
        </dataTable>
      </griddedTableDef>
    </functionDefn>
  </function>
```



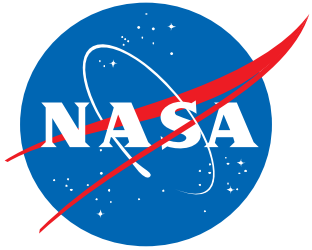
(assuming $ALP = [0:5:35]$)



Current DAVE-ML limitations

- Static models (dynamics not yet fully supported)
- No submodels (recursion not yet supported)
- No native DAVE-ML editor yet
 - Implies hand-editing with generic text or XML editor

DAVE-ML is at a sufficient stage of advancement to encode inertia, gear, and aero models, some of the more detailed subsystem models in flight simulation



Summary

- DAVE-ML 'ready-for-use' on production models
 - static models only, such as aero, mass properties
- Limited tools available, but slowly growing
 - Chicken or egg problem right now
 - Hand-coding not recommended - write export script
- Syntax will evolve in backwards-compatible way
- Still to tackle: dynamics & subsystems
- Help wanted! Join the mailing list - see website

<http://www.daveml.org>