The *DaveMLTranslator*

An Interface for DAVE-ML Aerodynamic Models

Melissa A. Hill — *Unisys Corp., Hampton, Virginia*
E. Bruce Jackson — *NASA Langley Research Center, Hampton, Virginia*

Overview

- Motivation
- DAVE-ML exchange format
- LaSRS++ framework
- Class Design & Implementation
- Operation of Translator
- Test Results
- Summary

Motivation

- Weeks or months to host or rehost simulation model for piloted or analysis of aerospace flight dynamic vehicle models
- Add support for draft AIAA standard simulation format, DAVE-ML, to Langley standard real-time simulation framework, LaSRS++
- Demonstration of capability of translation (versus compilation) approach for piloted (I.e.*real-time*) simulation
- Serve as an additional validation of the DAVE-ML format for both air- and spacecraft aerodynamic models

Traditional Approach to Sim Rehosting

- Usually a manual, semi-automated process
- Begins with receipt of flight simulation model source code and data tables
  - Sometimes with documentation and checkcases
  - Originator-specific variable names
  - Axis conventions are usually understandable
  - Differences on locus of action of forces & moments
- Involves translation into local site's namespace and table format and adding wrapper code to insert into site framework, followed by extensive checkcase generation and comparison
- Can take weeks or months, depending on availability or creation of one-time scripts to assist in conversions

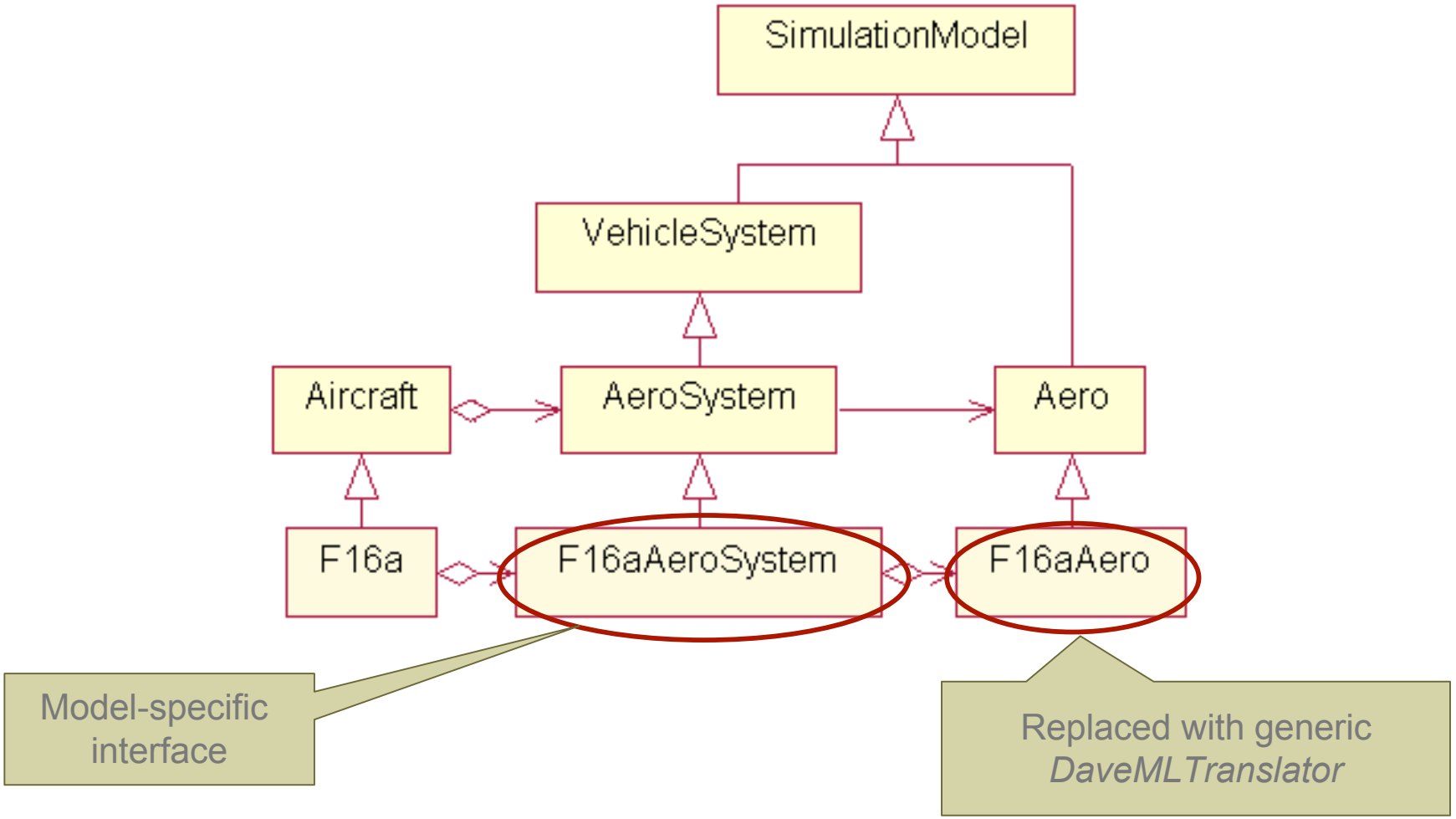
DAVE-ML Exchange Format

- Dynamic Aerospace Vehicle Exchange Markup Language, developed by AIAA Modeling & Simulation Technical Committee, uses a custom XML grammar. [Jackson & Hildreth, 2002]
- Sufficient for static (e.g. aero and inertia) subsystem models
- A DAVE-ML model usually contains:
  - Data tables (for linear or higher-order interpolation, any dimension)
  - Buildup equations (using standard W3C MathML grammar)
  - History or provenance of data (source docs, authors, modification history)
  - Uncertainty parameters / statistics of data (including cross-correlation)
  - Verification checkcases (including tolerances per parameter)
- See: http://daveml.nasa.gov

LaSRS++ Framework

- Langley Standard Real-Time Simulation in C++ (LaSRS++) [Leslie, Geyer, et al., 1998]
- C++ based, object-oriented, simulation framework used at NASA Langley for piloted simulations and off-line analysis of single and multiple vehicle experiments.
- LaSRS++ includes linear table interpolation class which is reused here.
- LaSRS++ has a standard *VehicleSystem* interface class to decouple aero, inertia, etc. models from the vehicle equations of motion (sits between vehicle and aero model); we use inheritance for each of these aero models (F-16, HL-20).
- Existing C++ F-16, HL-20 models are available for comparison

LaSRS++ Standard Class Hierarchy



DaveMLTranslator Class Design

DaveMLTranslator …

- Serves to interpret any static model written in DAVE-ML format
- Relies on existing LaSRS++ table interpolation utility class library
- Relies on new MathML interpretation class library
- Independent of the particular model being loaded
- Performs parsing, ordering, and verification of subsystem models

Operation

- Loading and validation
- XML element handling
- MathML equation handling
- Execution order sorting
- Model verification

Operation: Loading and Validating

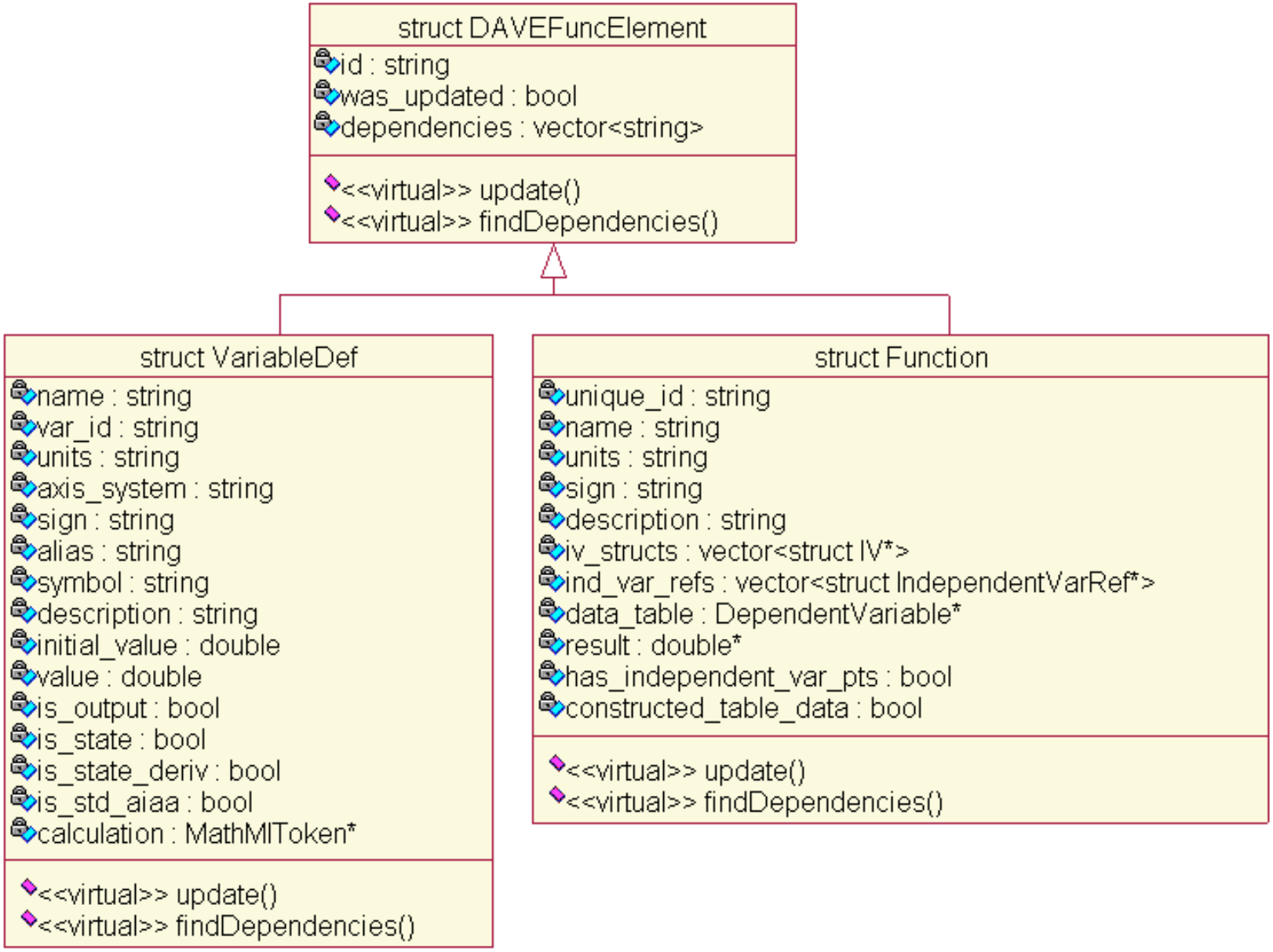
- The DAVE-ML model is loaded using open-source ***libxml2*** parser library into memory as a large tree of structures, called a Document Object Model
- ***libxml2*** also performs validation of grammar against the DAVE-ML Document Type Definition (ensuring no obvious errors such as missing or mis-ordered elements)

Operation: XML Element Handling

- Memory-resident DOM structure is then traversed, calling handler methods for each element encountered.
- Handler methods create more complete C++ structures suitable for real-time interpretation

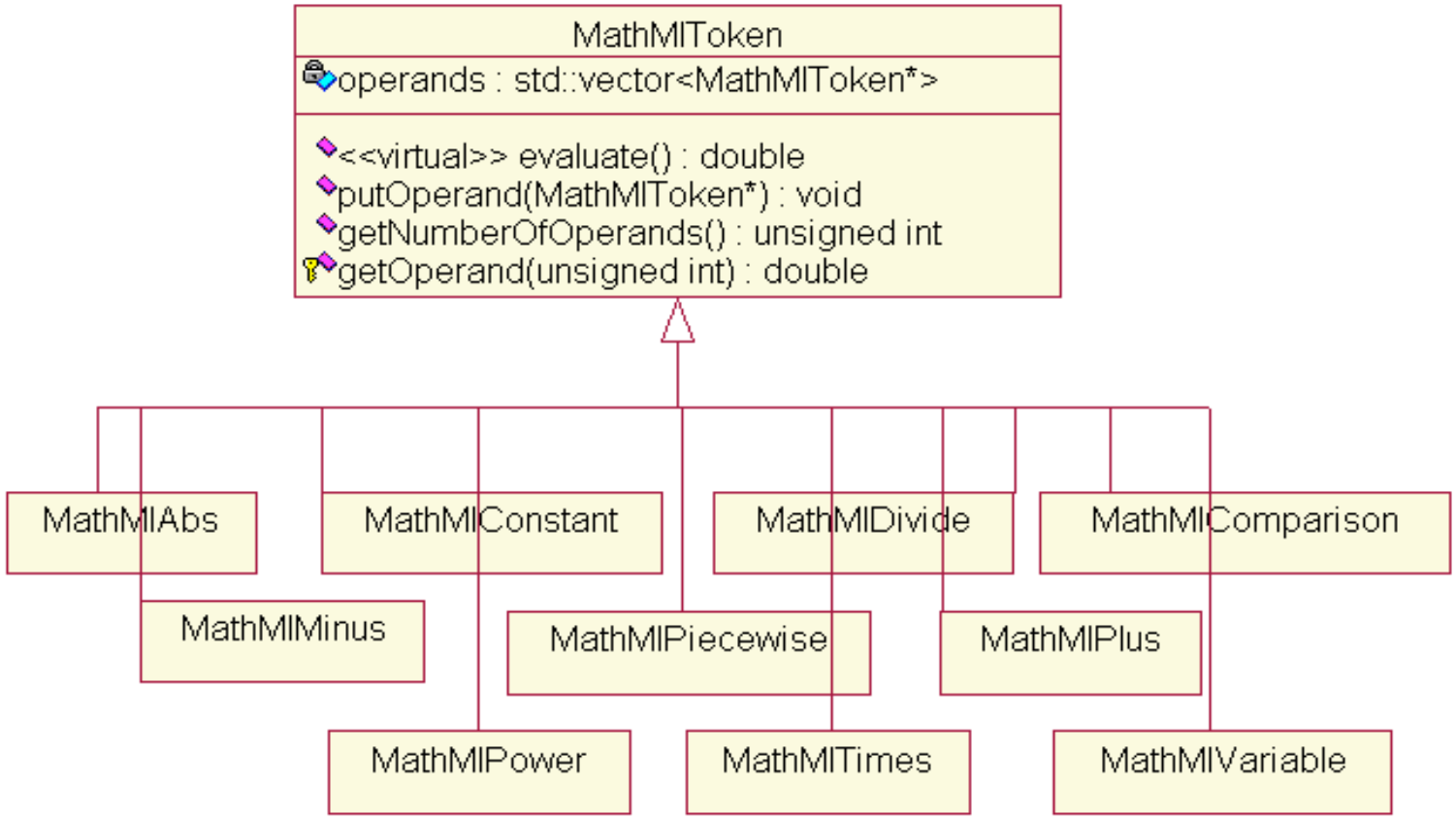
  Main types are *VariableDefs, Functions* and *MathML token*s

Operation: Element Handling (cont'd)



Operation: MathML Equation Handling

- MathML elements are created in memory as they are encountered in the DOM tree traversal
- A minimal set of MathML elements were defined to satisfy F-16 and HL-20 model requirements; others can be added as required



MathML Equation Handling (cont'd)

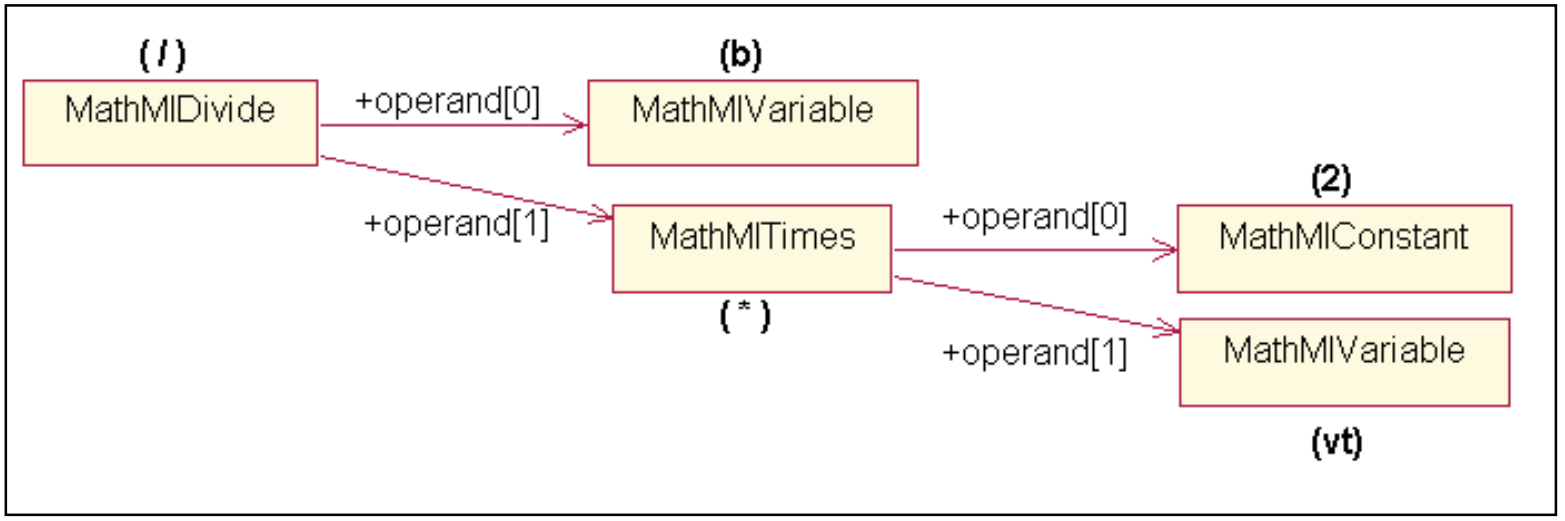
- MathML uses prefix notation:

$$\frac{b}{2V}$$

is encoded in MathML as

```
<math>
  <apply>
    <divide/>
    <ci>b</ci>
    <apply>
      <multiply/>
      <cn>2</cn>
      <ci>V</ci>
    </apply>
  </apply>
</math>
```

- In DaveMLTranslator structure:



Operation: Resolving Execution Order

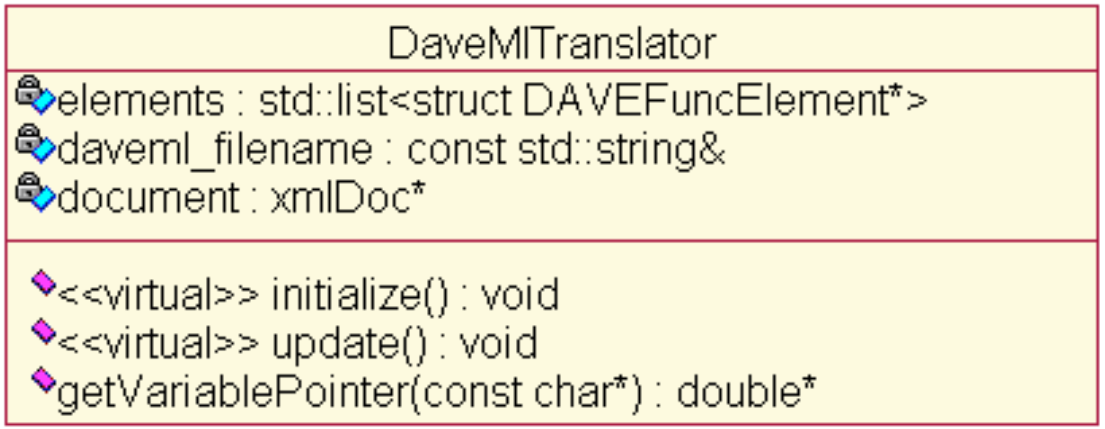
- DAVE-ML allows forward referencing, where calculations of variables defined early in the DAVE-ML model depend on the output of function table interpolations, defined later; the function tables typically depend on independent variables defined earlier.
- Circular references can't be ordered and are detected and flagged (indicating an unrealizable DAVE-ML model).
- After *variables*, *functions* and *math* elements are built, they are ordered into an execution list (all done during initialization).
- Subsequent operation is by this predefined execution order.

Operation: Model Verification

- Final step before beginning real-time operation is model verification.
- If checkcases are defined in DAVE-ML, they are run and results tested against defined tolerance.
- DAVE-ML checkcases consist of input/output vector pairs with tolerance
  Checkcase may include intermediate (internal) variables for debugging.
- If checkcases are successful, model reports ready for real-time operation.

Integration into LaSRS++

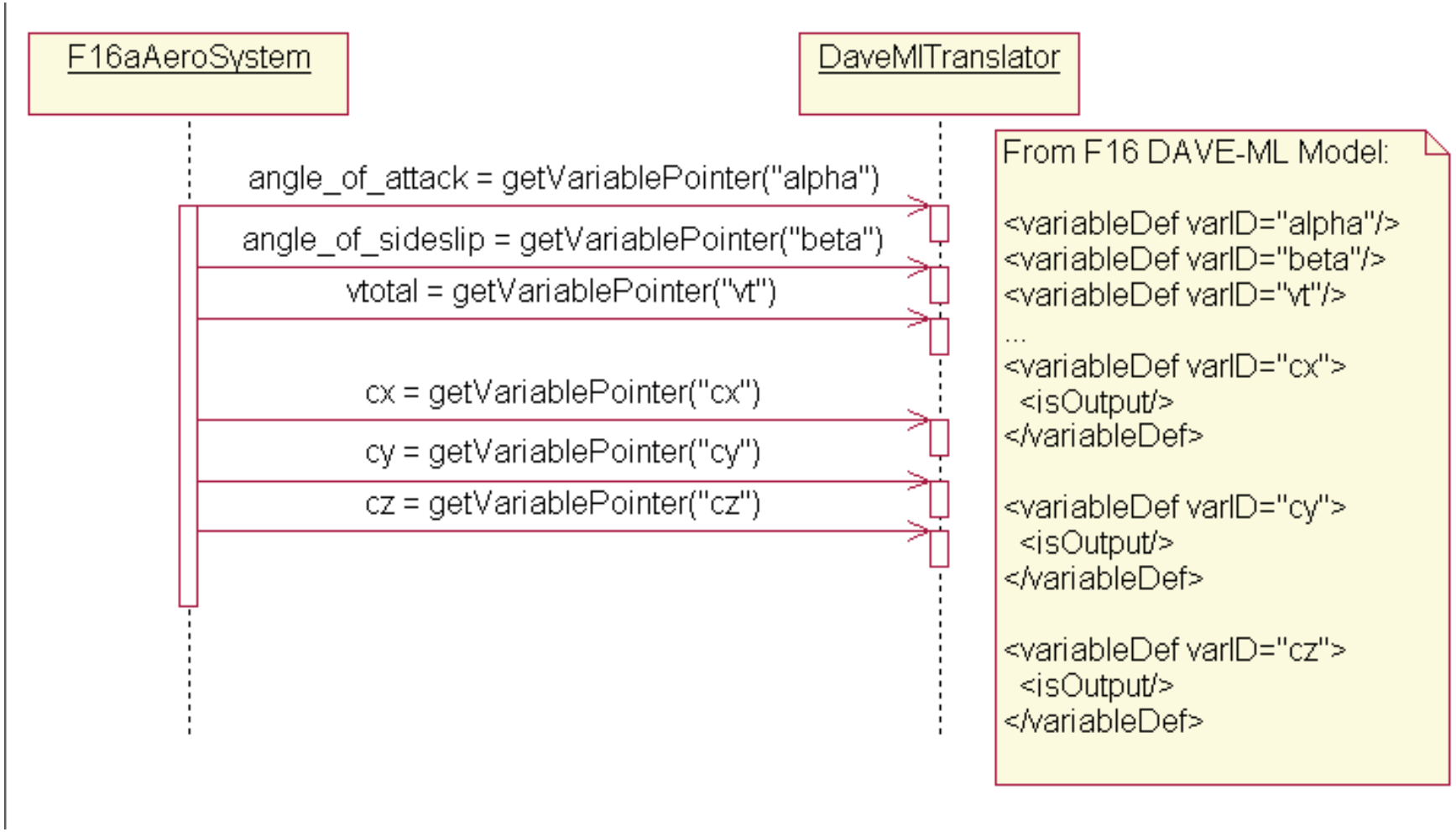
- *DaveMLTranslator* has a simple interface to *VehicleSystem:*



- **initialize()** resets variable values.
- **update()** executes the model.
- **getVariablePointer()** provides access to inputs and outputs of model.

Integration into LaSRS++ (cont'd)

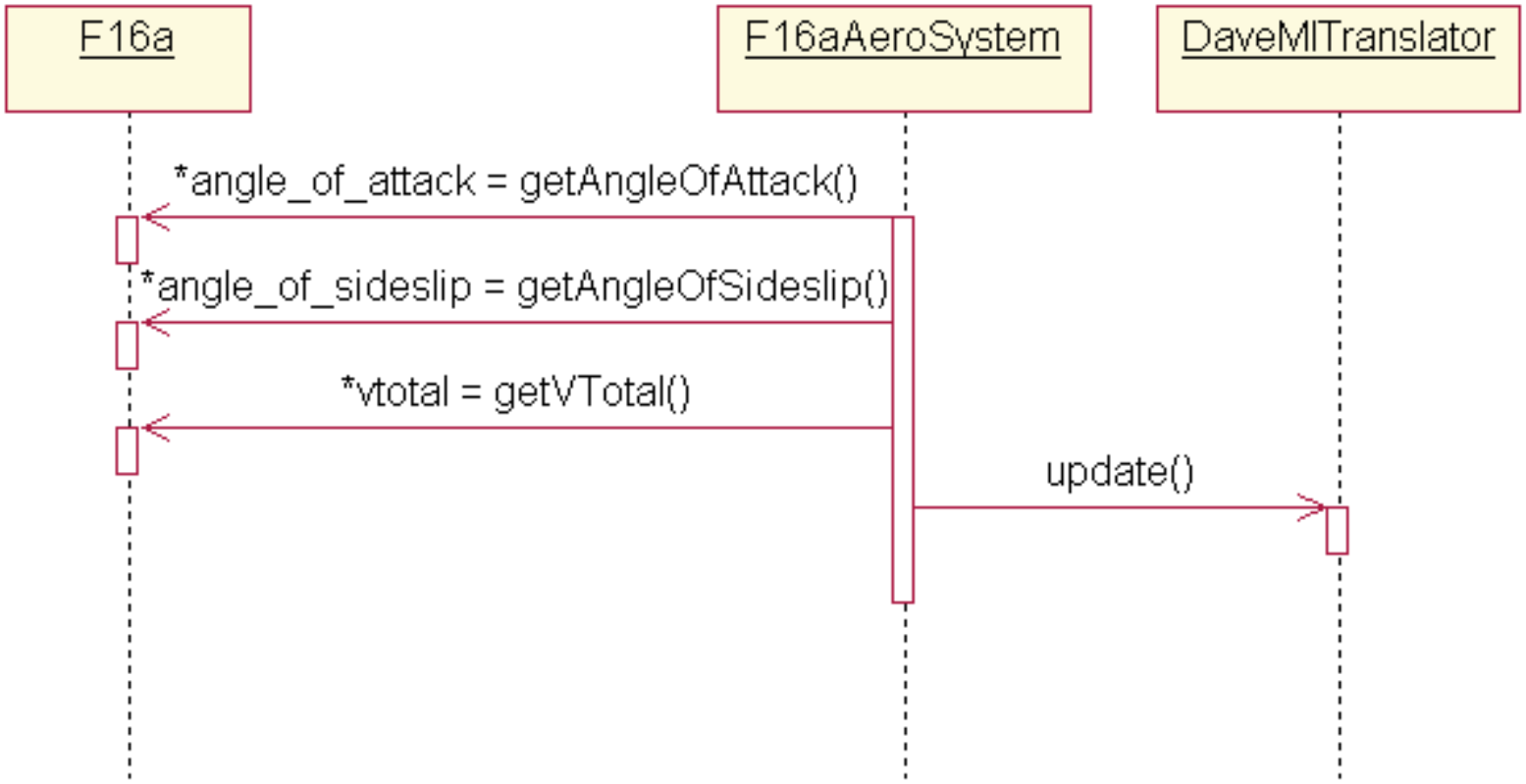
- At simulation initialization, the *VehicleSystem* requests pointers to inputs and outputs from it's *DaveMLTranslator:*



- *VehicleSystem* is aircraft-specific; *DaveMLTranslator* is generic for any model.

Integration into LaSRS++ (cont'd)

- During simulation execution, only **update()** is called by *VehicleSystem* after it obtains latest input values from the parent aircraft:



Test Results

- A comparison was made of the interpreted DAVE-ML models with C++ compiled models of

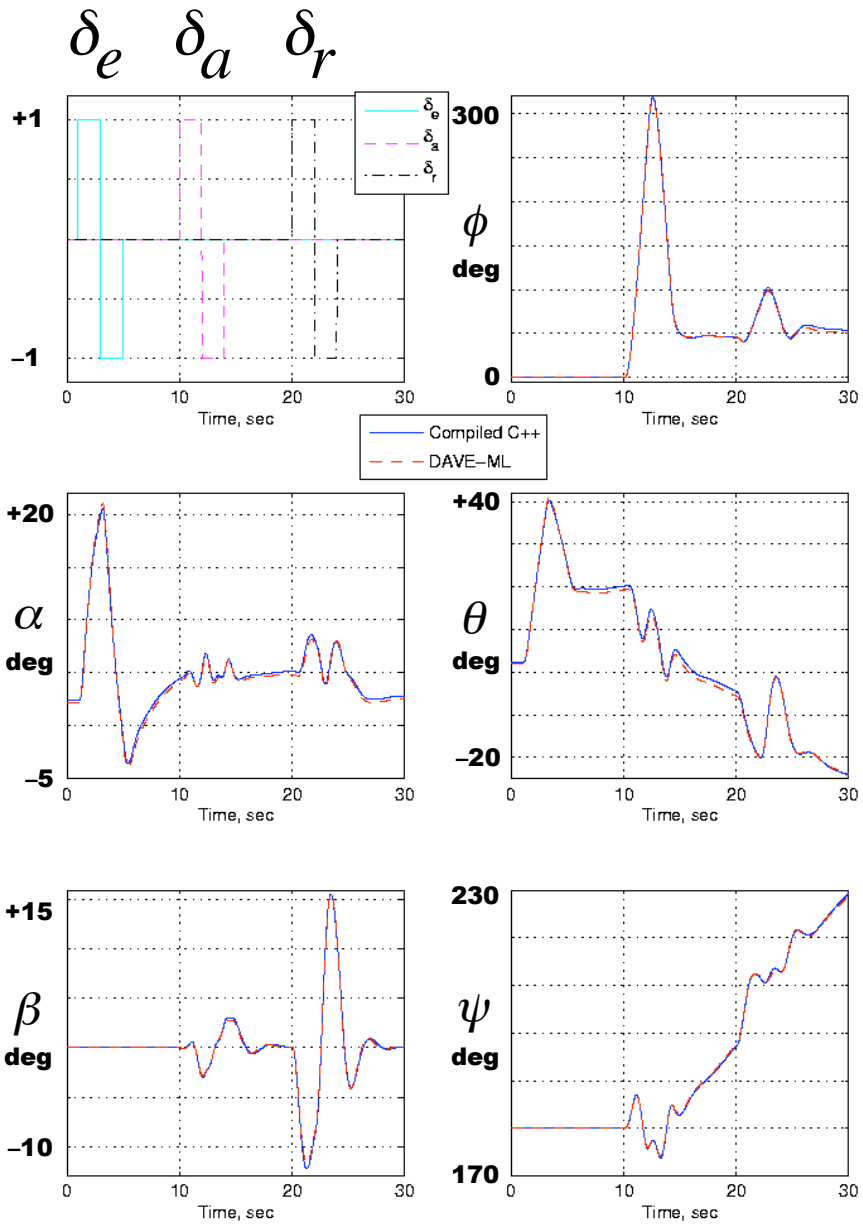
  F-16A subsonic aerodynamics

  HL-20 lifting-body Mach 0 - 4 aerodynamics

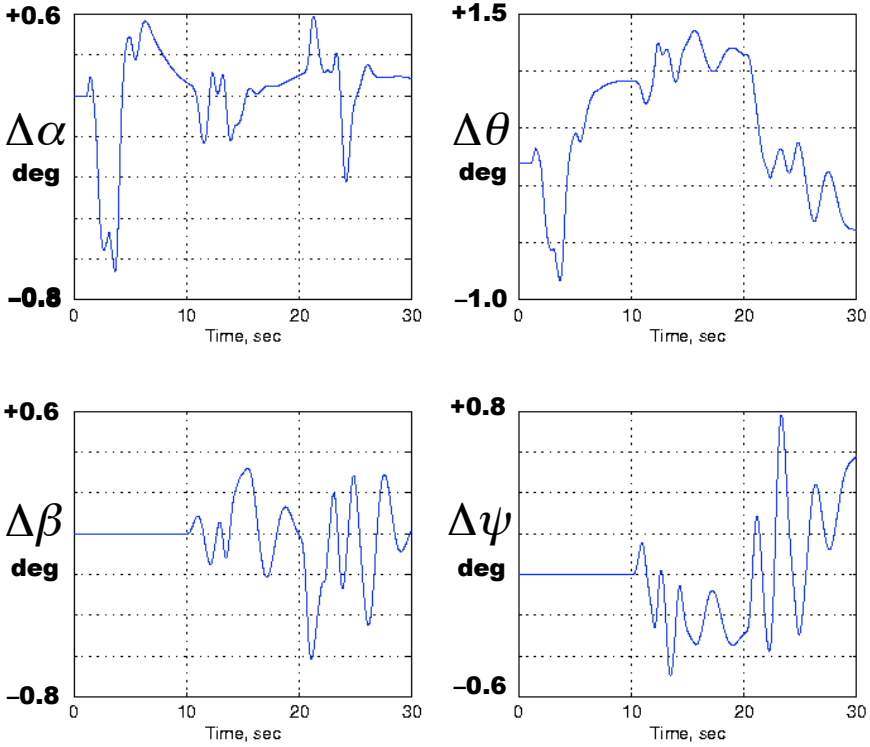
- F-16A models based on same wind tunnel test report (NASA TP-1538) but different math models.
- HL-20 models both based on same math model report (NASA TM-107580).

Test Results: F-16A Subsonic Aero

Comparisons

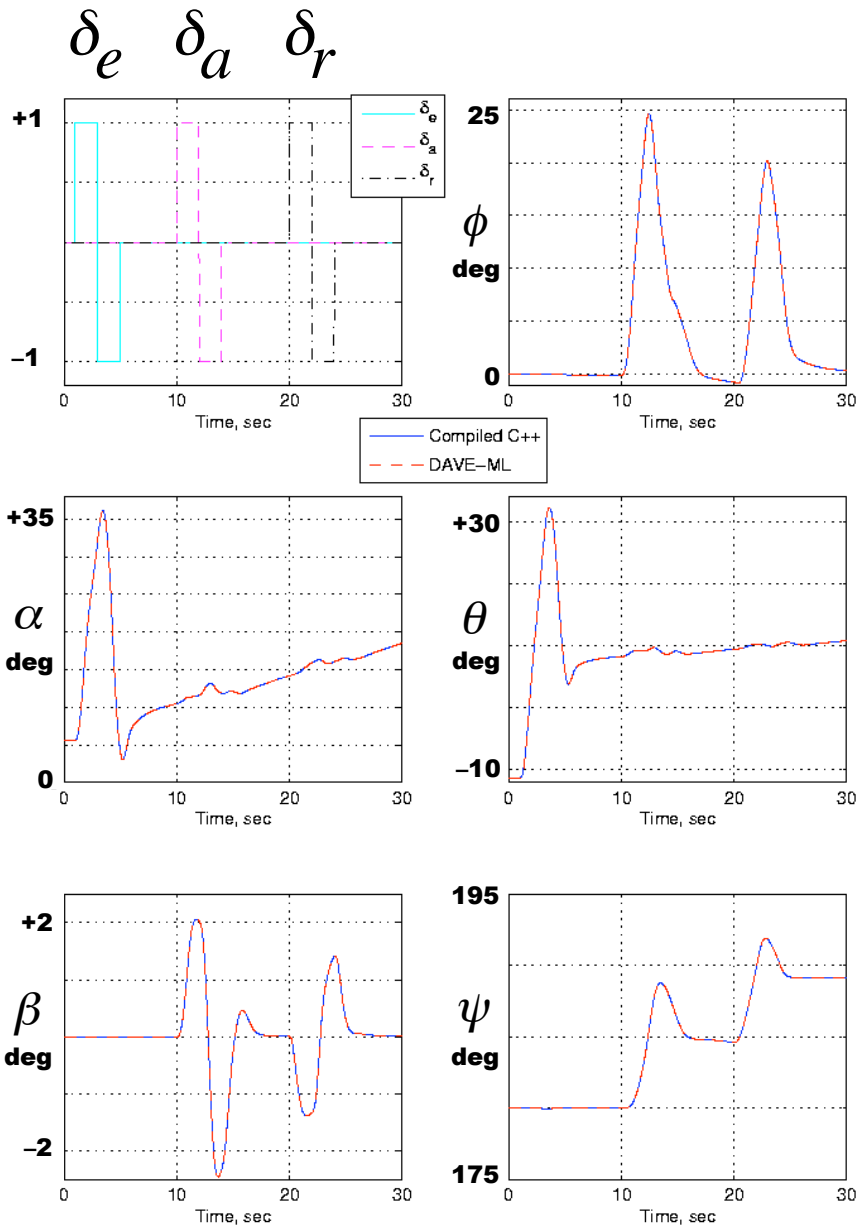


Residuals



Test Results: HL-20 Lifting Body Aero

Comparisons



Residuals

Test Results: Timing Data

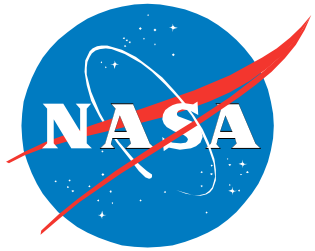
Single-frame Computation Time **ms**	**F-16A Subsonic Aero**			**HL-20 Aero**		
	C++	**DAVE-ML**	**Change**	**C++**	**DAVE-ML**	**Change**
Minimum	0.225	0.236	+4.9%	0.223	0.392	+76%
Mean	0.234	0.245	+4.7%	0.231	0.402	+74%
Maximum	0.248	0.266	+7.3%	0.301	0.472	+57%

Qualitative Piloted Evaluation

- In addition to time history comparisons, an informal piloted evaluation of both the F-16 and HL-20 simulations was conducted in the Langley Generic Flight Deck simulator.
- No obvious difference in models was apparent; both versions appeared to fly the same.

Summary

- A translator utility class, *DaveMLTranslator*, was written in C++ to load, validate, and verify DAVE-ML models at run-time in LaSRS++.
- The LaSRS++ table interpolation library was reused but the *DaveMLTranslator* is otherwise independent of other LaSRS++ class libraries.
- *DaveMLTranslator* is generic and model-independent.
- Changes to aircraft subsystem interfaces are required for I/O wiring.
- Time history comparison shows good match with C++ F-16 and identical match to C++ HL-20 aerodynamic models.
- The performance impact for an interpolated model is 5% to 75% slower than a compiled model.


NASA