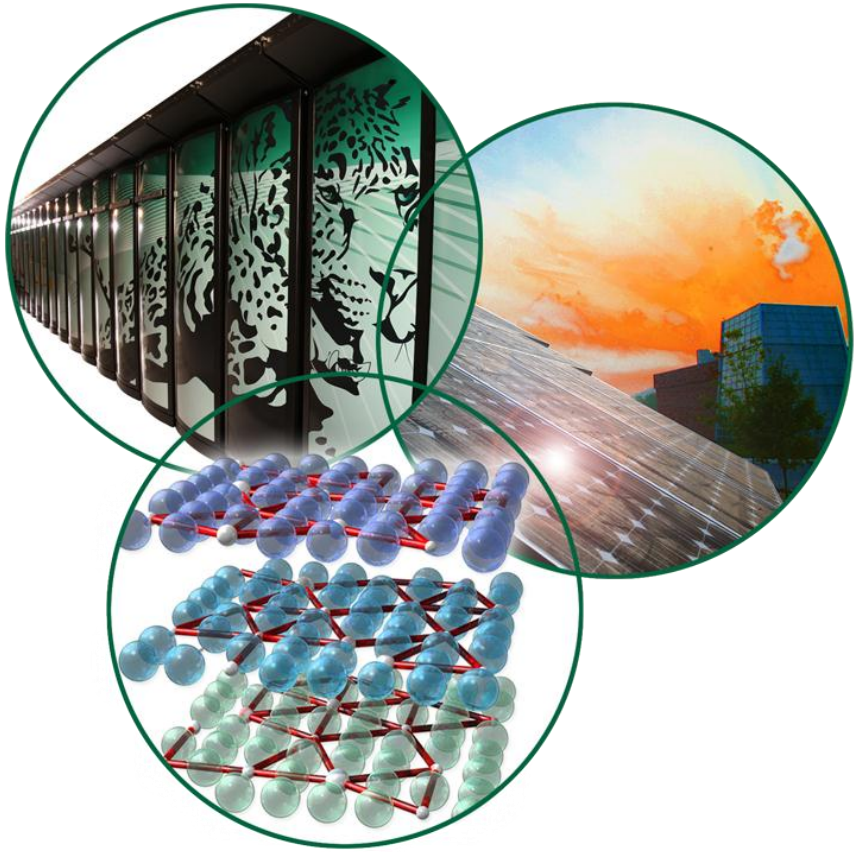
CAEBAT Open Architecture Software (OAS): Standardizing Input and Output

ORNL OAS Team



Expectations for today

- **Agree on the general framework/philosophy**
- **Provide input so that we can revise the presentation and send out a consensus report**
- **Form working groups (at least one person from each of the teams, ORNL, and NREL)**
  - Work out the details
  - Report back to the team in 6-8 weeks

Input

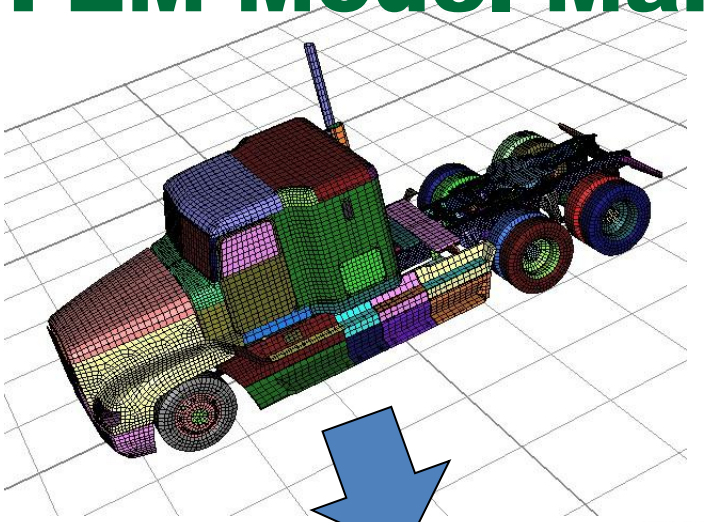
- **Exploit the hierarchical nature of batteries**
- **Agree on a standard description that works across different software platforms**
- **Common set of tools to process, visualize, and analyze the input data**
- **Translators to hook up to standard CAD packages for easy generation of CFD mesh**
- **XML would be a great choice as it is widely adopted, many third-party tools, etc.**
  - Can lead to interactive web-based capability….

XML Data Model

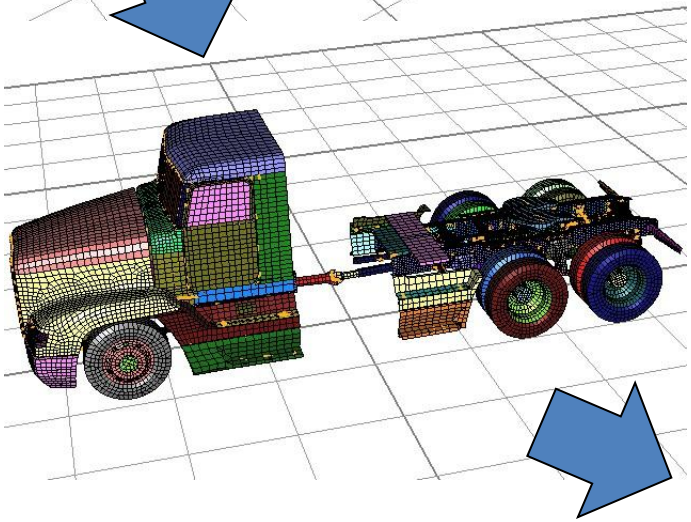
- Unifies the system around the common data model
- Presentation is separated from the data
- Allows for independent system development, XML can be directly served to templates and to the end user
- A large number of XML tools is available in public domain, no need to reinvent them.
- FEM model formats have evolved into hierarchical structured languages that map well into XML
- XML templates allow us to transform hierarchical data from one format to another.
- Existing XML technology can be efficiently applied to engineering problems.

Courtesy: Srdjan Simunovic

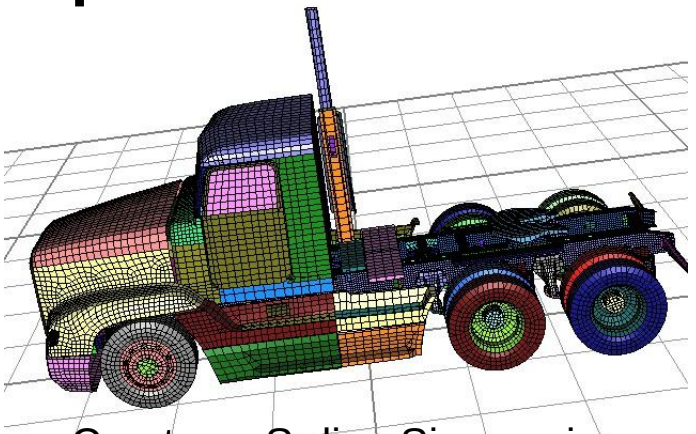
Example: Development of Tools for FEM Model Manipulation



- We use existing FEM models to generate new models for different impact scenarios
- We automate modifications in unified data system using transformation scripts.



- This reduces development time and allow for parametric studies



A day cab model from a sleeper cab model

http://thyme.ornl.gov/FHWA/TractorTrailer

Courtesy: Srdjan Simunovic

Hierarchical description of a battery

- Materials
- Components: Anode, Cathode, Separator, Electrolyte, Current Collector, Tab, Insulator, Cooling, ….
- Cell sandwich: as a composition of above along with geometric information
- Cell: Prismatic or wound (spiral/flat) using above descriptors and additional geometric information
- Modules: Composed of cells and additional components such as Cooling, Insulator,….
- Battery pack: Composed of modules, geometric specification and additional components such as cooling, insulator, …..

XML

```
<MAT_ANODE tag1="from XY" tag2="paper XYZ">
    <VAR>
      <MID>1001</MID>
      <RO>3.89e-09</RO>
      <k>2461</k>
      <alpha>0.323</alpha>
    </VAR>
    <DATAXY>
    OCP data in XY format…. (can be in functional form too)
    </DATAXY>
    <ADDON>
      <DESC>Anode material is graphite from XY company and described in XYZ paper</DESC>
     </ADDON>
</MAT_ANODE>

<MAT_CATHODE>, <MAT_ELECTROLYTE>, ……

<COMPONENT_ANODE tag1="from XY" tag2="paper XYZ">
    <VAR>
      <MID> 1001 </MID>
      <CID>1001</CID>
      <DX>3.89e-04</DX>
    </VAR>
    <DATAXY>
    OCP data in XY format…. (can be in functional form too)
    </DATAXY>
    <ADDON>
      <DESC>Anode material is graphite from XY company and described in XYZ paper</DESC>
    </ADDON>
</COMPONENT_ANODE>

<COMPONENT_CATHODE>……..

<CELL_SANDWICH_XYZ1> …..
<CELL_PRISMATIC1> …..
<CELL_SPIRAL1> ….
<MODULE_PRISMATIC> ……
<MODULE_PRISMATIC> ……
<MODULE_PRISMATIC> ……
<PACK_XYZ1> ……
<PACK_XYZ2> …….
```

Similar description can be standardized for other inputs

- **Model parameters**
  - Model choices (e.g. dual foil vs. lumped)
  - Different parameters and constants
  - Reactions (e.g., side reactions, SEI resistance)
- **Numerical/Simulation parameters**
- **Run parameters (e.g., cycle profiles)**

Battery state file

- **This file(s) will have minimal set of variables so that all the components can talk to each other and the state is completely defined**
- **We have, *for now*, chosen netCDF format for this file**
  - internal specifics for CAEBAT to be determined by community (definitions, variable names, underlying grid, etc.,)

Species Conservation

Electrolyte phase:

$$\frac{\partial(\varepsilon_e c_e)}{\partial t} = \nabla \cdot \left(D_e^{eff} \nabla c_e\right) + \frac{1-t_+^0}{F} j^{Li} - \frac{i_e \cdot \nabla t_+^0}{F}$$

Solid phase:

$$\frac{\partial(\varepsilon_s c_s)}{\partial t} = \nabla \cdot \left(D_s^{eff} \nabla c_s\right) - \frac{j^{Li}}{F}$$

Closures:

$$D_e^{eff} = D_e \varepsilon_e^{\xi} \qquad D_s^{eff} = D_s \varepsilon_s^{\zeta}$$

$$\frac{\partial(\rho c_p T)}{\partial t} = \nabla \cdot (\lambda \nabla T) + q$$

Charge Conservation

Electrolyte phase:

$$\nabla \cdot \left(\kappa^{eff} \nabla \phi_e\right) + \nabla \cdot \left(\kappa_D^{eff} \nabla \ln c_e\right) + j^{Li} = 0$$

Solid phase:

$$\nabla \cdot \left(\sigma^{eff} \nabla \phi_s\right) - j^{Li} = 0$$

Closures:

$$\kappa^{eff} = \kappa \varepsilon_e^{1.5} \qquad \kappa_D^{eff} = \frac{2RT\kappa^{eff}}{F}(t_+^0 - 1)\left(1 + \frac{d \ln f_{\pm}}{d \ln c_e}\right) \qquad \sigma^{eff} = \sigma \varepsilon_s^{m}$$

Electrode Kinetics

$$\bar{j} = a i_0 \left[\exp\left(\frac{\alpha_a F}{RT}\eta\right) - \exp\left(-\frac{\alpha_c F}{RT}\eta\right)\right]$$

Some features we like about netCDF format

- Support transient data (infinite dimension for time), units (very important for batteries)
- Broadly used in scientific computing community
- Lot of utility and tools for data analysis and visualization
- Parallel IO support

Issues:
- Unstructured data
- Mesh-to-Mesh translations (tets, hexes, polygons)
- Generalized mapping software
- Other standard file formats (HDF5, CGNS)

netCDF file

```
VAAYU:battery_state% ncdump cphit.nc | more
netcdf cphit {
dimensions:
        X = 10 ;
        Y = 10 ;
        Z = 101 ;
        time = UNLIMITED ; // (2 currently)
variables:
        float X(X) ;
                X:units = "m" ;
        float Y(Y) ;
                Y:units = "m" ;
        float Z(Z) ;
                Z:units = "m" ;
        float time(time) ;
                time:units = "sec" ;
        float concentration_solute(time, X, Y, Z) ;
                concentration_solute:units = "mol/l" ;
        float concentration_solid(time, X, Y, Z) ;
                concentration_solid:units = "mol/l" ;
        float potential_matrix(time, X, Y, Z) ;
                potential_matrix:units = "volts" ;
        float potential_solute(time, X, Y, Z) ;
                potential_solute:units = "volts" ;
        float temperature(time, X, Y, Z) ;
                temperature:units = "kelvin" ;
        float current_flux(time, X, Y, Z) ;
                current_flux:units = "A/m^2" ;
data:

 X = 0, 0.02666667, 0.05333333, 0.08, 0.1066667, 0.1333333, 0.16, 0.1866667,
    0.2133333, 0.24 ;

 Y = 0, 0.01555556, 0.03111111, 0.04666667, 0.06222222, 0.07777778,
    0.09333333, 0.1088889, 0.1244444, 0.14 ;
```