Model Integration with Python Wrapper and CSDMS

Harut Shahumyan
August 5, 2015
<table>
<thead>
<tr>
<th>Model</th>
<th>Environment</th>
<th>Operation System</th>
<th>Developer/Licensing</th>
<th>Simulation Period</th>
<th>Sim. years</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSTM</td>
<td>CUBE</td>
<td>Windows</td>
<td>Scripts: Open source CUBE: CitiLabs</td>
<td>2007 or 2030</td>
<td>1</td>
<td>15-16 hour</td>
</tr>
<tr>
<td>SILO</td>
<td>Java</td>
<td>Multi-platform</td>
<td>Open source</td>
<td>2007-2030</td>
<td>23</td>
<td>4-5 hour</td>
</tr>
<tr>
<td>MEM</td>
<td>CUBE</td>
<td>Windows</td>
<td>EPA (MOVES) / CitiLabs</td>
<td>2007 or 2030</td>
<td>1</td>
<td>&lt; 30 min</td>
</tr>
<tr>
<td>BEM</td>
<td>R</td>
<td>Multi-platform</td>
<td>n/a</td>
<td>2007 or 2030</td>
<td>1</td>
<td>&lt; 30 min</td>
</tr>
<tr>
<td>CBLCM</td>
<td>C / C++</td>
<td>CentOS</td>
<td>USGS</td>
<td>2007-2030</td>
<td>4</td>
<td>3 hour</td>
</tr>
</tbody>
</table>
Data flow between the models

- **CBLCM**
  - Land Cover Model
  - Population
  - Employment
  - Accessibility by auto & transit

- **SILO**
  - Land Use Model
  - Building data: type, age, area, rooms, occupation, heating fuel, location, etc.

- **MSTM**
  - Transport Model
  - Population
  - Employment
  - Auto availability

- **MEM**
  - Mobile Emission Model
  - All trips within the region
  - Average speed distribution

- **BEM**
  - Building Emission Model
  - Auto travel time
  - Transit travel time
  - Auto-operating costs
Processing flow order and simulation periods

- **2000-2012**
  - SILO

- **2012**
  - MSTM
  - MEM
  - BEM
  - CBLCM

- **2012-2040**
  - SILO

- **2040**
  - MSTM
  - MEM
  - BEM
  - CBLCM
Key Requirements of Integration

- Ability to develop models independently, such that they may be plugged-in easily.
- A modular approach supporting reusability and adding new components.
- User friendly graphical interface.
- Minimizing manual data transfer.
- Minimal or no change in source codes of the models.
- Capacity to link models developed in different programming languages and environments.
- Ability to deal with different licensing requirements.
- Compatibility with GIS for easy data visualization and spatial analysis.
- Minimal costs and efficient timing for implementation.
Coupling Methodologies

- **tool coupling**: framework provides tools to support embedded and integrated models, single GUI, common data storage
- **joined coupling**: one model embedded in other or two in parallel, single GUI, common data storage
- **shared coupling**: single GUI and separate data storage, or multiple GUIs and common data storage
- **loose coupling**: modeler interfaces with each model, uses automated data transfer
- **one-way data transfer**: modeler interfaces with each model, manually transfers data

Python Wrappers: Integration Status

<table>
<thead>
<tr>
<th>Models</th>
<th>Current Data Exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQM</td>
<td>Water Quality Model</td>
</tr>
<tr>
<td>CBLCM</td>
<td>Land Cover Model</td>
</tr>
<tr>
<td>CWRF</td>
<td>Climate Change Model</td>
</tr>
<tr>
<td>INFORUM</td>
<td>Economic Model</td>
</tr>
<tr>
<td>SILO</td>
<td>Land Use Model</td>
</tr>
<tr>
<td>BEM</td>
<td>Building Emission Model</td>
</tr>
<tr>
<td>MSTM</td>
<td>Transport Model</td>
</tr>
<tr>
<td>MEM</td>
<td>Mobile Emission Model</td>
</tr>
</tbody>
</table>

**Loose Coupling System**
Developed through Python & ArcGIS Model Builder

- **Installed and coupled on the designated server**: Automatic (loose coupling)
- **Installed on different remote servers or PCs**: Manual (coupling is under development)
- **To be determined**
Benefits

- No need to change the source codes of the models.
- Runs models developed in different environments.
- Can be extended with additional models over time.
- General user interface showing process flow.
- Rich visualisation & mapping capabilities with ArcGIS.
- Easy to implement.

Limitations

- Parallel model runs and dynamic data exchange during simulation time steps are not supported.
- Model processes run independently from one another.
- Data exchanged between modules are written to and read from a hard drive. No in-memory data exchange.
Coupling Methodologies

- **tool coupling**: framework provides tools to support embedded and integrated models, single GUI, common data storage.
- **joined coupling**: one model embedded in other or two in parallel, single GUI, common data storage.
- **shared coupling**: single GUI and separate data storage, or multiple GUIs and common data storage.
- **loose coupling**: modeler interfaces with each model, uses automated data transfer.
- **one-way data transfer**: modeler interfaces with each model, manually transfers data.

• Open Modelling Interface (OpenMI)
• Community Surface Dynamics Modeling System (CSDMD)
• Earth System Modeling Framework (ESMF)
• Model Coupling Toolkit (MCT)
• O-PALM
• OASIS
• FLUX
• Kepler
Capacity building, Community networking

Members from 500+ Institutions (academic, government, industry) from 68 countries

<table>
<thead>
<tr>
<th>Working Groups</th>
<th>Focus Research Groups</th>
<th>Initiatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terrestrial</td>
<td>Hydrology</td>
<td>Coastal Vulnerability 40</td>
</tr>
<tr>
<td>Coastal</td>
<td>Carbonate</td>
<td>Continental Margins 25</td>
</tr>
<tr>
<td>Marine</td>
<td>Geodynamics</td>
<td></td>
</tr>
<tr>
<td>EKT</td>
<td>Chesapeake</td>
<td></td>
</tr>
<tr>
<td>Cyber</td>
<td>Critical Zone</td>
<td></td>
</tr>
<tr>
<td>Human Dimension</td>
<td>Anthropocene</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ecodynamics</td>
<td></td>
</tr>
</tbody>
</table>

Hydrology 470
Carbonate 80
Geodynamics 80
Chesapeake 60
Critical Zone 50
Anthropocene 40
Ecodynamics 20

Source: James Syvitski

http://csdms.colorado.edu
• Applied to simplify conversion of an existing model to a reusable, plug-and-play model component.

• A set of functions with prescribed function names, argument types and return types.

• BMI functions are straightforward to implement in any of the languages supported by CSDMS (C, C++, Fortran, Java and Python).

• Maps input and output variable names to CSDMS Standard Names.

• Any model that provides the BMI functions can be easily converted to a CSDMS plug-and-play component.
BMI restructuring requirements are implemented in SILO

```java
void initialize (in string config_file)
void update (in double dt)
void finalize ()
void run_model (in string config_file)
```

SILO is made available at GitHub

CSDMS Standard Naming for SILO is under development
Next steps

• **SILO**
  • Finalize the variable standard naming for SILO
  • Provide SILO code to CSDMS development team to converted it to a CSDMS plug-and-play component.

• Based on the lessons learned proceed with **BEM** and **CBLCM** models.

• **MSTM** and **MEM** require different approach because of CUBE restrictions