Space Weather Modeling Framework
Component Interface Workshop

David Chesney and Ovsei Volberg
Software Engineering
Center for Space Environment Modeling
The University of Michigan
Objective

The SWMF provides a flexible Sun-Earth simulation framework serving the Sun-Earth community.

In its fully developed form, the SWMF will comprise a series of interoperating models of physics domains, ranging from the surface of the Sun to the upper atmosphere of the Earth.
Approach

Historic: Tightly Coupled Code
Approach

Present: Framework w/ Small Number of Components
- Prototype Phase
- Milestone 7I, Aug03
Approach

Future: Full Framework

- Final Phase
- Milestone 10J, Jul04
Physics Component (Language)

Physics **Component** is:
- **Physics Model**
  - Solution code for physics domain
  - Language: F77, F90
- **Physics Model Wrapper**
  - Unit conversion
  - Data transformation
  - Language: same as Physics Model
- **Physics Model Interface**
  - Data exchange between physics modules
  - Completes mappings between two coupled physics modules
  - Receives and processes signals from CON
  - Monitors the execution of coupled physics modules
  - Language: 7I: F90; 10J: F90, C++
# SWMF (Language)

<table>
<thead>
<tr>
<th>SWMF</th>
<th>Description</th>
<th>Language:</th>
<th>Language:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics Model Interface</td>
<td>See previous slide</td>
<td>F90</td>
<td>C++/F90</td>
</tr>
<tr>
<td>Control</td>
<td>• Distribution of PMs to PEs</td>
<td>F90</td>
<td>C++/F90</td>
</tr>
<tr>
<td>GUI</td>
<td>• Choice of PMs</td>
<td>HTML, XML, Javascript, PERL</td>
<td>Same</td>
</tr>
<tr>
<td>Output File (OUF)</td>
<td>• Output file format</td>
<td>Proprietary</td>
<td>Proprietary, HDF-5</td>
</tr>
</tbody>
</table>
Language Independence

Language neutrality? Yes and No

- Yes:
  - Follow OOP paradigms
  - Consistent definition of functions/methods:
    - constructor
    - destructor
    - get
    - set
    - solve
    - stop
Language Independence

Language neutrality? Yes and No

- No:
  - Phase 7I written in F90
## System Requirements (Execution)

<table>
<thead>
<tr>
<th>Machine</th>
<th>Location</th>
<th>OS</th>
<th>Fortran Compiler</th>
<th>C/C++ Compiler</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beowulf (16 PEs)</td>
<td>CSEM – UMich</td>
<td>RedHat Linux v7.2</td>
<td>NAG v4.0a</td>
<td>gcc/g++</td>
<td>execution</td>
</tr>
<tr>
<td>Beowulf (120 PEs)</td>
<td>CSEM – UMich</td>
<td>RedHat Linux v7.2</td>
<td>NAG v4.0a</td>
<td>gcc/g++</td>
<td>execution</td>
</tr>
<tr>
<td>SGI Origin (16 PEs)</td>
<td>CAC – UMich</td>
<td>IRIX 6.5</td>
<td>MIPSpro 7</td>
<td>MIPSpro 7</td>
<td>execution</td>
</tr>
<tr>
<td>SGI Origin (1024 PEs)</td>
<td>NASA – Ames</td>
<td>IRIX64</td>
<td>MIPSpro</td>
<td>MIPSpro</td>
<td>execution</td>
</tr>
<tr>
<td>Compaq</td>
<td>NASA – GSFC</td>
<td>Tru64 Unix</td>
<td>f90</td>
<td>cxx</td>
<td>execution</td>
</tr>
</tbody>
</table>
# System Requirements

(Development and GUI)

<table>
<thead>
<tr>
<th>Machine</th>
<th>Location</th>
<th>OS</th>
<th>Fortran Compiler</th>
<th>C/C++ Compiler</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Athalon</td>
<td>CSEM – UMICH</td>
<td>RedHat Linux v7.2</td>
<td>NAG v4.0a</td>
<td>gcc/g++</td>
<td>development</td>
</tr>
<tr>
<td>(2 PEs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMD Athalon</td>
<td>CSEM – UMICH</td>
<td>RedHat Linux v7.2</td>
<td></td>
<td></td>
<td>GUI</td>
</tr>
<tr>
<td>(1 PE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Varies</td>
<td>Varies</td>
<td></td>
<td></td>
<td></td>
<td>remote user</td>
</tr>
</tbody>
</table>

Machine: remote user
Execution Model

- Milestone 7I: *Sequential – Parallel* model of execution
  - Physics modules executing on multiple processors (i.e., parallel), however, only one physics module is executing at any given time (i.e., sequential)

- Milestone 10J: *Sequential – Parallel* model of execution acceptable, however, *Concurrent – Parallel* model encouraged
  - Physics modules executing on multiple processors (i.e., parallel), with multiple physics modules executing at same time (i.e., concurrent)
In general:
- Component/Component
  - peers
  - cannot spawn each other
  - no side effects

However:
- CON/Component
  - client/server
Persistence

- Component (Physics Model) Persistence
  - fixed at compile time

- Compute Resource (Processing Element) Persistence
  - fixed at compile time
Standards
(The Wild Frontier between Responsibility and Authority)

- **Interoperability Document**
  - ✔ Initialized in parallel configuration.
  - ✔ Activated by Control Module for data exchange with other component.
  - ✔ Queried for run status, output, and input data.
  - ✔ Perform time synch and conversion, and performance profiling.
  - ✔ Provide error trapping and report to the Control Module.
  - ✔ Write state to a restart file.
  - ✔ Read input data from files (path to files is externally controlled/set).
  - ✔ Write output data to files (path to files is externally controlled/set).
  - ✔ To be registered by the Control Module.

- **Data Naming Standard**
Target User Community

- Modelers in space physics community and space weather forecasters
Target Component Authors

Modelers in the space physics community

**Phase 7I**
- Inner magnetosphere model: Rice University

**Phase 10J**
- Energetic particle model: University of Arizona
- Eruptive event generator: University of New Hampshire
- Radiation belt model: Rice University
- Kinematic heliosphere model: UCSD
Summary

- Implementation of framework based upon phased development
- Milestone 7I (Prototype Phase) due in August, 2003
  - decouple existing code
  - recouple with standard interfaces
  - centralize control
  - show ability to substitute alternative physics components
- Milestone 10J (Final Phase) due in July, 2004
  - add many additional physics components