## An Update on Software Developments in the FATES

**Ryan Knox (rgknox@lbl.gov), Charles Koven, Benjamin Andre, Rosie Fisher, Gautam Bisht, Mariana Vertenstein, William Riley, William Sacks, Erik Kluzek and David Lawrence**

### An Interface Between FATES and a Host Land Model (HLM)

**Interface?**
Software that allows modules (e.g., FATES and the HLM) to communicate.

**Why do we need a (good) interface?**
1. enable an easier connection of FATES with other/new models/modules
2. buffer FATES engineers and scientists from having to change/adapt our model when someone changes something in the host model
3. constrain and identify the flow of information/data into and out of our module

**V1 Design Factors:**
1. Create an isolated portion of code in the HLM where HLM-allowed and FATES-allowed data structures can interact
2. Create data structures in FATES specifically for input and output to a host model (boundary conditions, IO)
3. Use modern Fortran “class” structures which help to further protect how/where data are used and allow for object oriented programming
4. The FATES should be able to compile stand-alone and act as a library (if desired)
5. “Thread safe” design
6. A comprehensive and challenging suite of tests
7. Utilize the HLM’s “low level” IO software (instead of self)

### FATES Development At a Glance

- **Conceived December 2015**
- >22,000 lines of code added
- >15,000 lines of code removed
- 33 Separate Issues Have Been Completed
- 49 Separate Submissions have been integrated
- 7 People have contributed to code
- 595 commits (change groups) to the master branch
- 17 forks (personal copies) of the repository itself

### Milestones Completed

- Created a new repository from CESM trunk 121
- Unified pre-existing developments from NCAR/LBNL/LANL
- Ported code to lawrencium, wolf, conejo and PCLinux machines
- Added Linkages between ED and HLM soil BGC
- Numerous bug fixes
- Expanded diagnostics to included size/pft dimensioning and higher frequency flux variables
- Converted FATES memory structure - sites aligned to columns
- General structure of the HLM-FATES API
- General structure of the FATES public API
- Cold-start interface
- Partial completion of restart interface
- Interface for photosynthesis
- Interface for radiation
- Interface for hydraulics
- Interface for soil BGC
- Partial completion of interface for ED driver
- Interface for History IO diagnostics
- Interface/cleaning of phenology
- Interface communicator for global variables

### Milestones Until V1.0

- Complete interface to model restarting
- Interface and separate FATES parameter input
- Complete interface and boundary conditions to ED driver
- Final pass-through for CLM/CESM globals
- Naming conventions, documentation general code cleaning
- Refactor build and module control system
- Implement HLM-side interface into ACME

### Cross Project Development Liasing

**Call graph of HLM-FATES connections**

### Model Testing

**All requests to update the code are tested** on 2+ machines (NCAR-yellowstone, LBNL-lawrencium and periodically NCAR hobart) with a total of 4 compiler combinations (intel,gnu,pgi and NAG!). We have protocols for submitting and describing changes, and peer review.

**CIME (Common Infrastructure for Modeling the Earth) build and test system. PASS/FAIL software tests:**

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoke Tests</td>
<td>Evaluate user-level failures of the build process</td>
</tr>
<tr>
<td>Restart Tests</td>
<td>Assess the ability of the code to restart after a crash</td>
</tr>
<tr>
<td>Layout Change Tests</td>
<td>Test the layout of the code on different machines</td>
</tr>
<tr>
<td>FATES on/off</td>
<td>Test the ability of the code to be compiled and run on different machines</td>
</tr>
<tr>
<td>Module X on/off</td>
<td>Test the ability of the code to be compiled and run on different machines</td>
</tr>
</tbody>
</table>

### Model Science Testing

Some changes to the model should give identical results as its basis (regression tests), *sometimes changes will impact outcomes and need some basic scientific evaluation* and sanity testing.

**Toolset:** “rapid science checking (RSC) tool”, written in python, uses minimal libraries (matplotlib, netcdf), controlled via command-line arguments and an XML, designed for automated use. Evaluates time-series projections of flux, and forest structure and composition.