Outline

• CASL background
• What is LIME?
• Integration of RELAP5-3D into VERA
• Current status of RELAP5-3D in the CASL program
• Summary
Can an advanced “Virtual Reactor” be developed and applied to proactively address critical performance goals for nuclear power?

1. Reduce capital and operating costs per unit energy by:
   - Power uprates
   - Lifetime extension

2. Reduce nuclear waste volume generated by enabling higher fuel burnups

3. Enhance nuclear safety by enabling high-fidelity predictive capability for component and system performance from beginning of life through failure
CASL has selected key phenomena limiting reactor performance selected for challenge problems

<table>
<thead>
<tr>
<th>Phenomena</th>
<th>Power uprate</th>
<th>High burnup</th>
<th>Life extension</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operational</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUD-induced power shift (CIPS)</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>CRUD-induced localized corrosion (CILC)</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Grid-to-rod fretting failure (GTRF)</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Pellet-clad interaction (PCI)</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><strong>Fuel assembly distortion (FAD)</strong></td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><strong>Safety</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Departure from nucleate boiling (DNB)</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cladding integrity during loss of coolant accidents (LOCA)</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Cladding integrity during reactivity insertion accidents (RIA)</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><strong>Reactor vessel integrity</strong></td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reactor internals integrity</strong></td>
<td>✗</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The CASL Virtual Reactor (VERA) builds on a foundation of mature, validated, and widely used software.
What is LIME?

• An acronym for Lightweight Integrating Multi-physics Environment for coupling codes

• A tool for creating multi-physics simulation code(s) that is particularly useful when computer codes are currently available to solve different parts of a multi-physics problem

• One part of the larger VERA framework being developed in CASL
Important characteristics of LIME

• LIME is designed to:
  – Enable separate physics codes ("new" and "old") to be combined into a robust and efficient fully-coupled multi-physics simulation capability
  – Allow composition of both controlled and open-source components, enabling protection of export-controlled or proprietary code while still allowing distribution of the core system and open components

• LIME is not limited to:
  – Codes written in one particular language
  – A particular numerical discretization approach (e.g., finite element)

• LIME is not "plug and play":
  – Requires revisions/modifications to most stand-alone physics codes
  – Requires the creation of customized "model evaluators"
Key components of a simple generic application created using LIME
Revisions and modifications that may be required of a physics code

• Console I/O must be redirected (no pause statements or read/write to standard streams)

• Each code must be wrapped so the multi-physics driver can link to it (i.e., like a library)

• Each code must be organized into several key parts that can be called independently
  – Initialization: *read inputs, allocate memory*…
  – Solve: *compute solution for a given time step and state*
  – Advance: *copy converged state and prepare for next step*
Status of LIME

- Theory manual: Sandia report SAND2011-2195
- User manual: Sandia report SAND2011-8524
- LIME 1.0 (source and documentation) on sourceforge.net
Refactorization of stand-alone RELAP5-3D
Improvements to Model Evaluator

- Modifications needed to move from stand-alone to a coupled capability
- Further refactoring of RELAP5 to allow LIME to control time steps
  - R5solve split into three new routines
  - Corresponding function calls added to model evaluator
- LIME program manager needed to be modified to handle re-negotiation of time step size after RELAP5-3D cuts (or increases) it
**LIME time step control**

**LIME Problem Manager**

- `time_step = determine_time_step()`
- `set_time_step()`
- `solve_nonlinear(test_time_step)`
- `has_time_step_changed(test_time_step, time_step)`
- `current_time += time_step`
- `update_time()`

**RELAP5 Model Evaluator**

- `get_time_step()`
  - `return caslmod_mp_requested_dt`
- `set_time_step(double dt)`
  - `caslmod_mp_lime_dt_ = dt`
- `solve_standalone(double & dt)`
  - `RELAP5_R5TAKE_TIME_STEP_F77()`
  - `dt = caslmod_mp_lime_dt`
- `update_time()`
RELAP5_ModelEval.cpp

//----------------------------------------- constructor -----------------------------------------

RELAP5_ModelEval::RELAP5_ModelEval(const LIME::Problem_Manager & pm,
    const string & name,
    Epetra_Comm& relap5_sub_comm,
    const std::string& input_file,
    const std::string& output_file,
    const std::string& restart_file) :

    problem_manager_api(pm),
    m_my_name(name),
    timer(0),
    m_input_file(input_file),
    m_output_file(output_file),
    m_restart_file(restart_file)
{
    RELAP5_R5SETUP_F77(&input_file[0],
                      &output_file[0],
                      &restart_file[0],
                      input_file.length(),
                      output_file.length(),
                      restart_file.length());

    RELAP5_R5PRE_STEP_F77();
}
//---------------------------- destructor ------------------------------------------

RELAP5_ModelEval::~RELAP5_ModelEval()
{
    RELAP5_R5FINALIZE_F77 ();
}

//------------------------------- solve_standalone -----------------------------------

bool RELAP5_ModelEval::solve_standalone(double & dt)
{
    RELAP5_R5TAKE_TIME_STEP_F77 ();
    dt = caslmod_mp_lime_dt_;
    return (true);
}

//----------------------------- get_time_step ----------------------------------------

double RELAP5_ModelEval::get_time_step() const
{
    return caslmod_mpRequested_dt_;
Conversion of RELAP5-3D build system

• TriBITS (VERA build system) uses CMake
  – Cross-platform, open-source build system
  – Uses compiler-independent configuration files to generate native makefiles

• RELAP5-3D build scripts replaced by CMake files
  – Easier integration with TriBITS
  – Necessary for inclusion in CASL automated software testing
  – Allows out-of-tree builds
Addition of RELAP5-3D to CASL testing

- VERA software packages stored in CASL repository under Git revision control
- Automated testing checks out appropriate source, performs builds, and runs tests at various frequencies
  - **Check in test script**: manual process to do basic testing and determine if it is safe to commit/push changes
  - **Continuous integration**: continuous loop that runs tests when global repository changes are detected
  - **Nightly regression testing**: a range of VERA configurations are built and tested with different compilers (e.g., gnu and Intel)
- Emails sent to relevant developers when failures are detected
Role of RELAP5-3D in CASL

- VERA is being developed to address challenge problems
- Initial emphasis is on core physics/TH and crud deposition
Role of RELAP5-3D in CASL

• VERA Requirements Document describes technical abilities VERA should provide
  – *capability to integrate* systems analysis codes (e.g. RETRAN, RELAP5, RELAP7) to support performance of nuclear safety analyses and analysis of plant accidents and transients
    • RIA
    • LOCA
    • Non-LOCA transients and accidents
  – These capabilities to be added in stages as relevant challenge problems are addressed
• RELAP5-3D is currently on hold
**Future development issues**

- RELAP5-3D can’t be distributed with VERA
  - Export control
  - License issues
- Will CASL version be synced with INL development version? If so, how?
  - It must be, if RELAP5-3D is supplied by licensees
  - RELAP5-3D not maintained in an accessible repository
  - CASL costs associated with merging new RELAP5-3D version
  - INL costs associated with ongoing maintenance of CASL mods
- Software use agreement applies to version 3.0.0
- INL would have to maintain a VERA environment for QA testing
- What about training, support, etc.?
Summary

- Initial VERA integration completed early this year
- CASL continues to address current challenge problems
- Safety analysis challenge problems not yet defined
- Further development on hold for FY13
- Unanswered questions about ongoing/future maintenance issues