RELAP5 and CASL

Peter Cebull

July 27, 2011

Outline

- CASL overview
- What is LIME?
- Role of RELAP5-3D in CASL
- Initial RELAP5-3D Integration
- Recent Improvements
- 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:

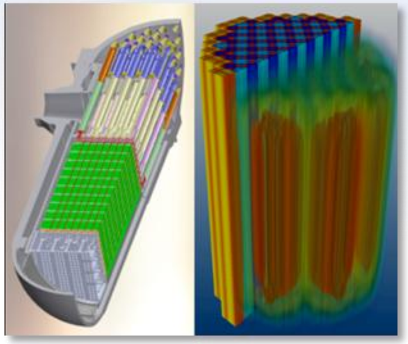
- Power uprates
- Lifetime extension

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

	Power uprate	High burnup	Life extension
Operational			
CRUD-induced power shift (CIPS)	×	×	
CRUD-induced localized corrosion (CILC)	×	×	
Grid-to-rod fretting failure (GTRF)		×	
Pellet-clad interaction (PCI)	×	×	
Fuel assembly distortion (FAD)	×	×	
Safety			
Departure from nucleate boiling (DNB)	×		
Cladding integrity during loss of coolant accidents (LOCA)	×	×	
Cladding integrity during reactivity insertion accidents (RIA)	×	×	
Reactor vessel integrity	×		×
Reactor internals integrity	×		×

CASL Organization

CASL Board of Directors
Ernest Moniz, Chairman

U.S. Department of Energy

Chief Strategy Officer
Mario Carelli

Doug Kothe, **Director**
Ronaldo Szilard, **Deputy**
Paul Turinsky, **Chief Scientist**

Partnership/Alliance Management
Jeff Cornett

Program Management
Jeff Banta
Collaboration & Ideation
April Lewis

Operations
Jayson Hines
Startup Manager
Gil Weigand

Materials Performance & Optimization
Chris Stanek
Sid Yip
Brian Wirth

Virtual Reactor Integration
John Turner
Randy Summers
Rich Martineau

Advanced Modeling Applications
Jess Gehin
Zeses Karoutas
Steve Hess

Models and Numerical Methods
Bill Martin
Rob Lowrie

Validation and Uncertainty Quantification
Jim Stewart
Dan Cacuci

COUNCILS

Science
John Ahearne, Chairman

Industry
John Gaertner, Chairman

Education
John Gilligan, Chairman

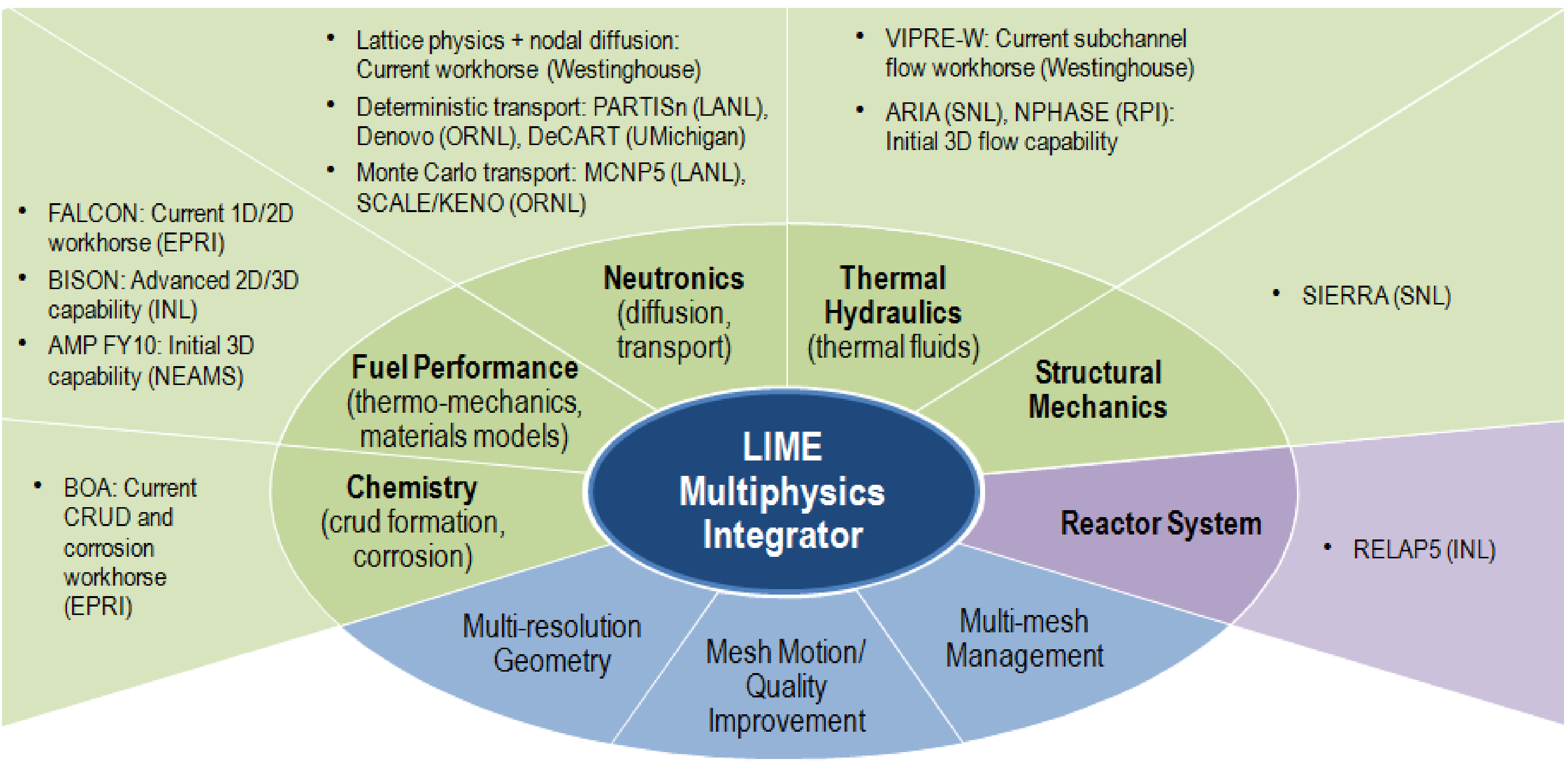
Commercialization
Jeff Cornett, Chairman

Communications, Policy, and Economic Development
Ken Nemeth, Chairman

CASL Senior Leadership

CASL Technical Leadership

The CASL VR (VERA) builds on a foundation of mature, validated, and widely used software



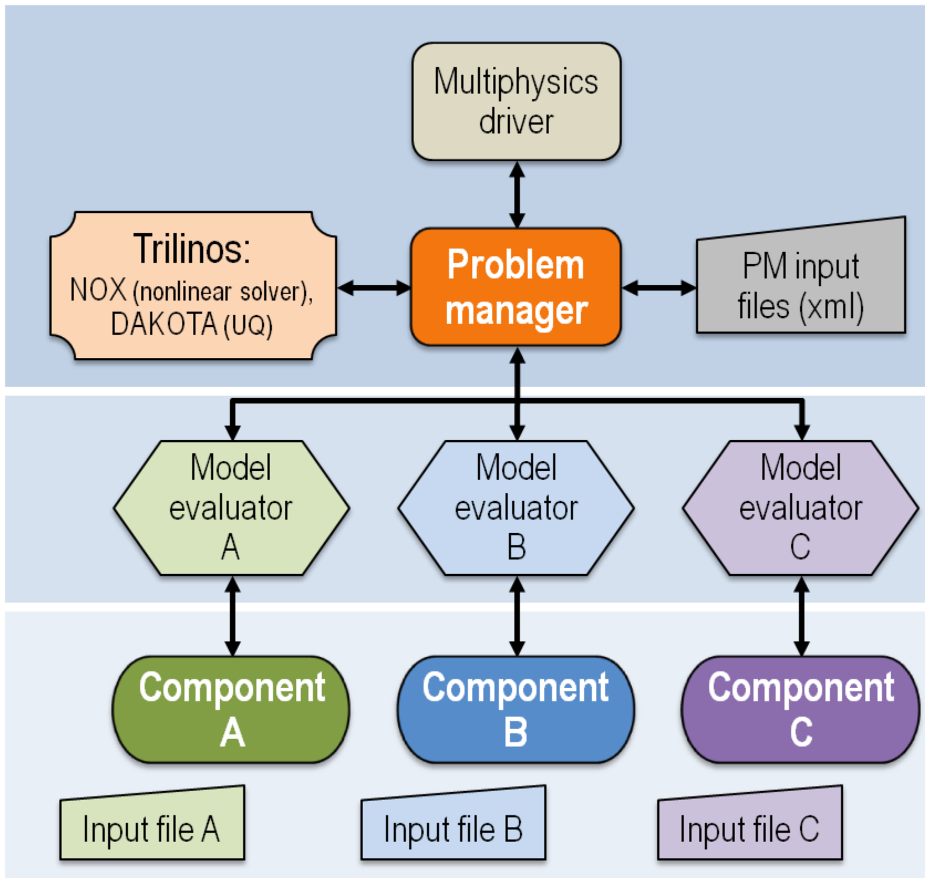
What is LIME?

- 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
- Intended to provide
  - Key high-level software,
  - A well-defined approach (including example templates),
  - And interface requirements for participating physics codes to enable assembly of these codes into a robust and efficient multi-physics simulation capability.
- 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

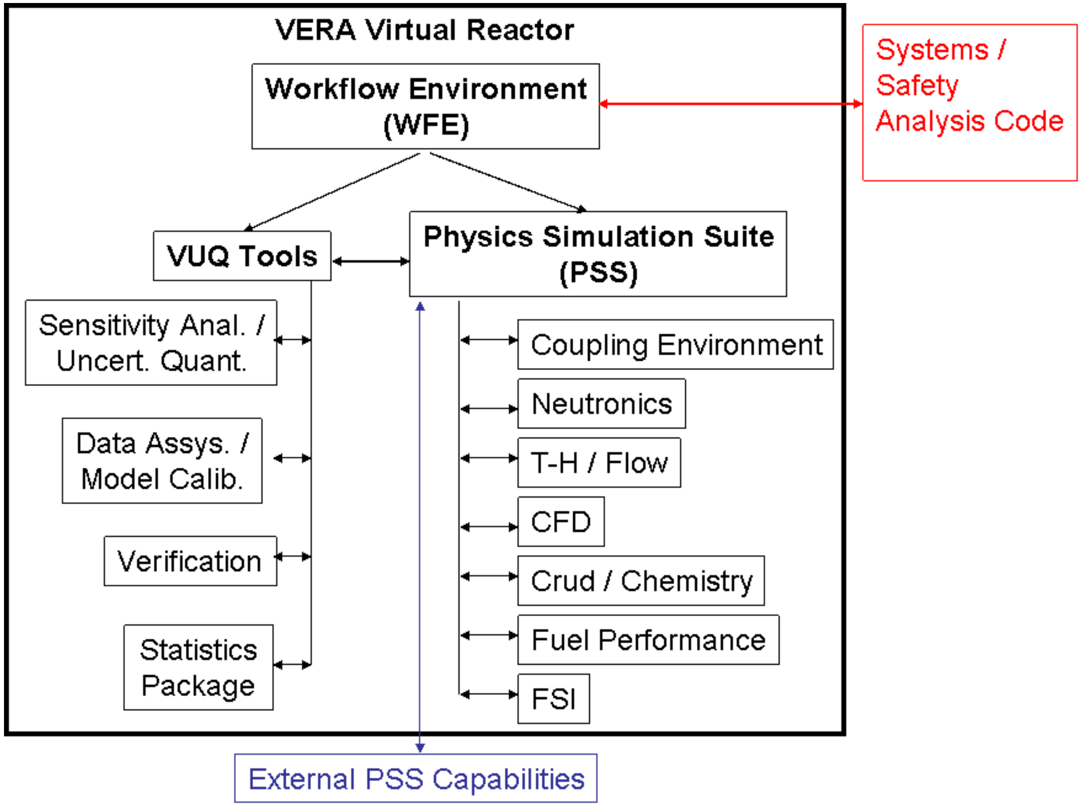
- Console I/O must be redirected (no pause statements or read/write to standard streams)
- Each code must be wrapped so the multi-physics driver can link to it (i.e., like a library)
- Each code must be organized into several key parts that can be called independently
  - Initialization: *read inputs, allocate memory…*
  - Solve: *compute solution for a given time step and state*
  - Advance: *copy converged state and prepare for next step*

Status of LIME

- Open source license being processed – being made available through Trilinos
- Theory manual just released: Sandia report SAND2011-2195
- User manual in draft form
- LIME is not a fully mature tool
  - Basic functionality exists and has been tested, but could benefit from review and optimization

Role of RELAP5-3D in CASL

- VERA is being developed to address challenge problems
- Initial emphasis is on core physics/TH and crud deposition



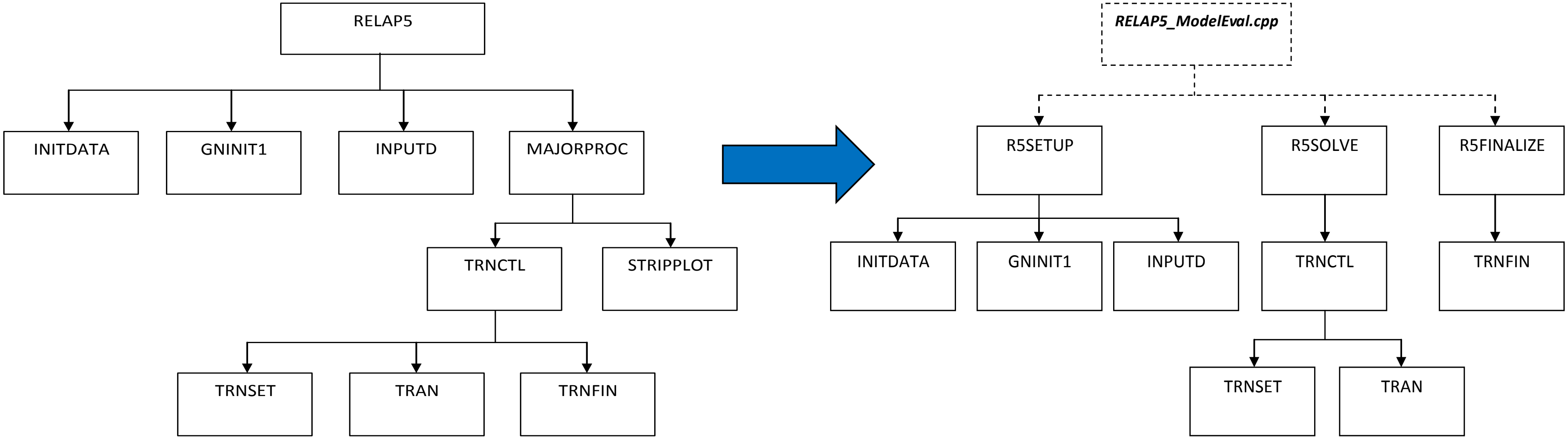
Role of RELAP5-3D in CASL

- VERA Requirements Document describes technical abilities VERA should provide
  - capability to integrate systems analysis codes (e.g. RETRAN, RELAP5, R7) 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 expected to play a larger role later (years 4/5?)

Initial Integration of RELAP5-3D

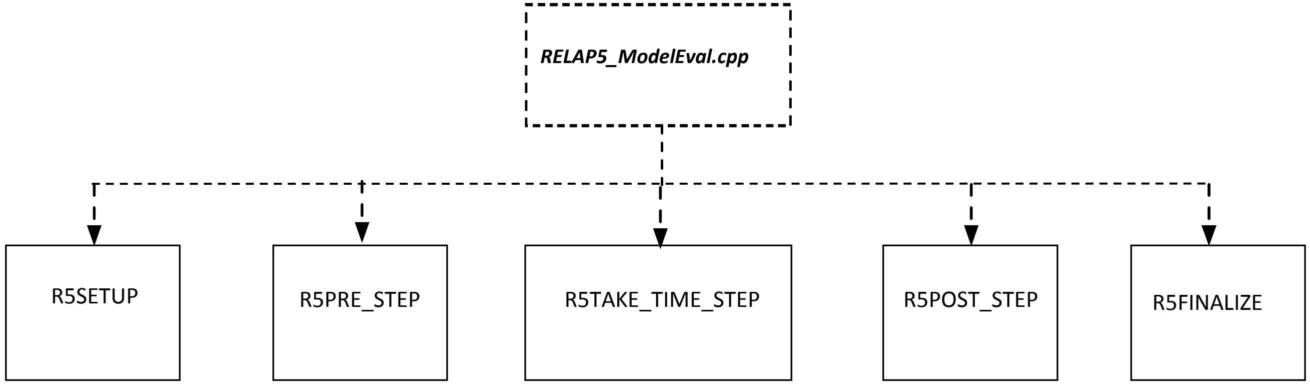
- Permission to give RELAP5-3D to CASL (r3d300casl) obtained 01/07/2011
- Modifications were made to run stand-alone under LIME
  - All writes to stdout ("tty") were redirected to a file
  - Code was refactored, three new subroutines added
  - Build scripts were modified to produce libraries instead of an executable
  - A CASL flag was added ("cr64") to conditionally implement the above changes (i.e., *`dinstls linuxntl cr64 nonpa`*)
- Stand-alone integration of RELAP5 completed 02/17/2011

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 needs to be modified to handle re-negotiation of time step size after RELAP5-3D cuts (or increases) it



RELAP5_ModelEval.cpp (1)

```
//---------------------- 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 ();
}
```

RELAP5_ModelEval.cpp (2)

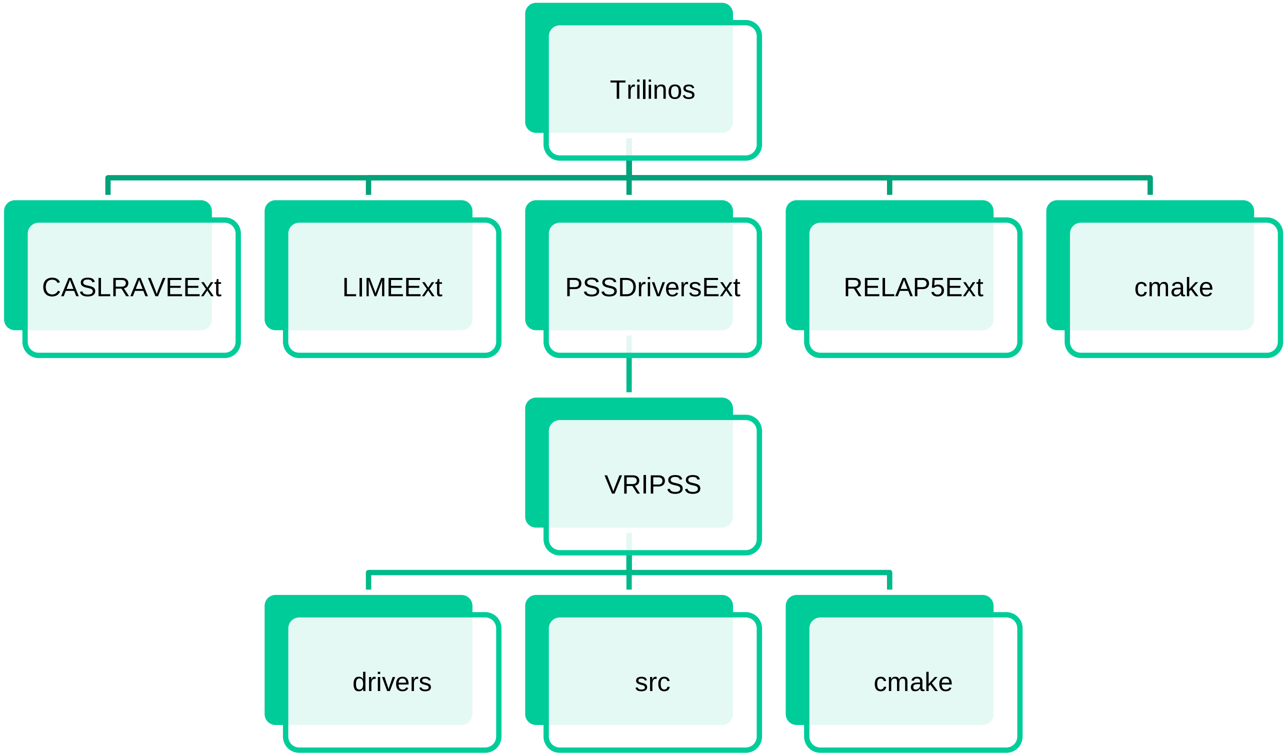
```
//---------------------- destructor ----------------------------------------

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

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

void RELAP5_ModelEval::solve_standalone()
{
  RELAP5_R5TAKE_TIME_STEP_F77 ();
}
```

RELAP5_ModelEval.cpp (3)

```
//----------------------- get_time_step -------------------------------------

double RELAP5_ModelEval::get_time_step() const
{
  return *ctrlmod_mp_dt_;
}

//----------------------- get_current_time ----------------------------------

double RELAP5_ModelEval::get_current_time() const
{
  return *ctrlmod_mp_timehy_;
}

//----------------------- update_time ---------------------------------------

void RELAP5_ModelEval::update_time()
{
  RELAP5_R5POST_STEP_F77 ();
}
```

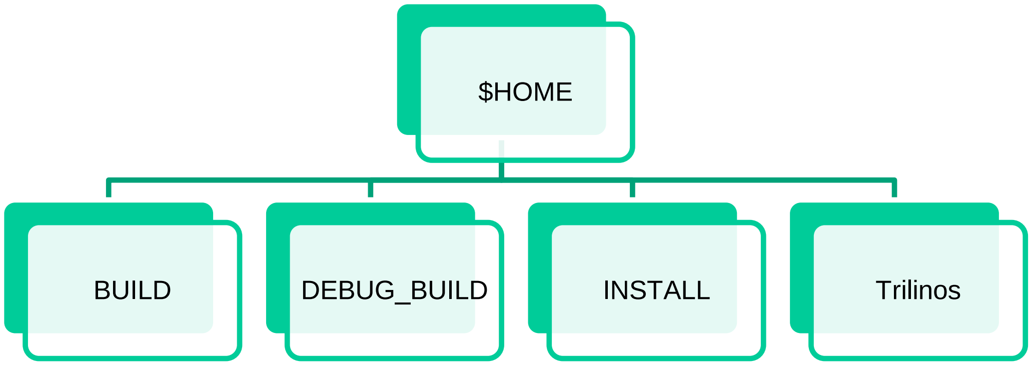
VERA and Trilinos

- VERA software is implemented as Trilinos external packages
- Physics codes are being converted to use Trilinos build system



Conversion of RELAP5-3D Build System

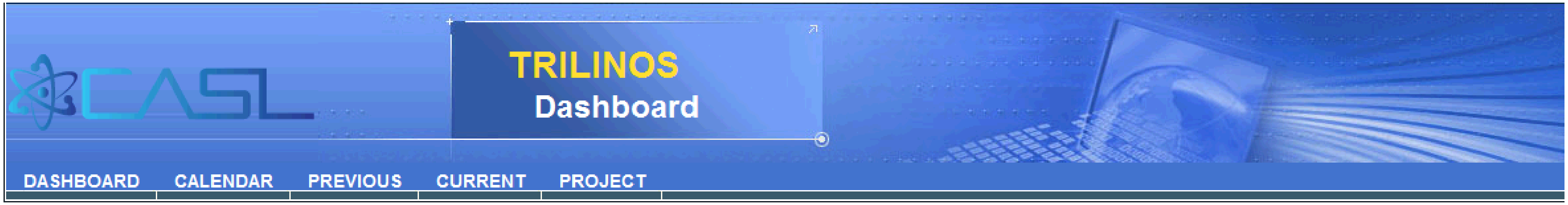
- Trilinos uses CMake
  - Cross-platform, open-source build system
  - Uses compiler-independent configuration files to generate native makefiles
- RELAP5-3D build scripts replaced by CMake files
  - Easier integration with Trilinos build system
  - Necessary for inclusion in CASL automated software testing
  - Allows out-of-tree builds



Addition of RELAP5-3D to CASL Testing

- RELAP5-3D CMake conversion allows inclusion in automated testing process
- 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

CASL CDash Dashboard



Project

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Trilinos	0	32	104	0	5	99	0	0	41

SubProjects

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
TrilinosFramework									
Teuchos	0	0	4	0	0	4			
ThreadPool	0	4	4	0	0	4			
VRIPSS	0	0	4	0	0	4	0	0	4
STARCCM	0	4	4	0	0	4			
DeCART	0	4	4	0	0	4	0	0	9
SEACAS									
CASLRAVE	0	3	3	0	3	0	0	0	3
CASLBOA	0	5	5	0	0	5	0	0	5

Summary

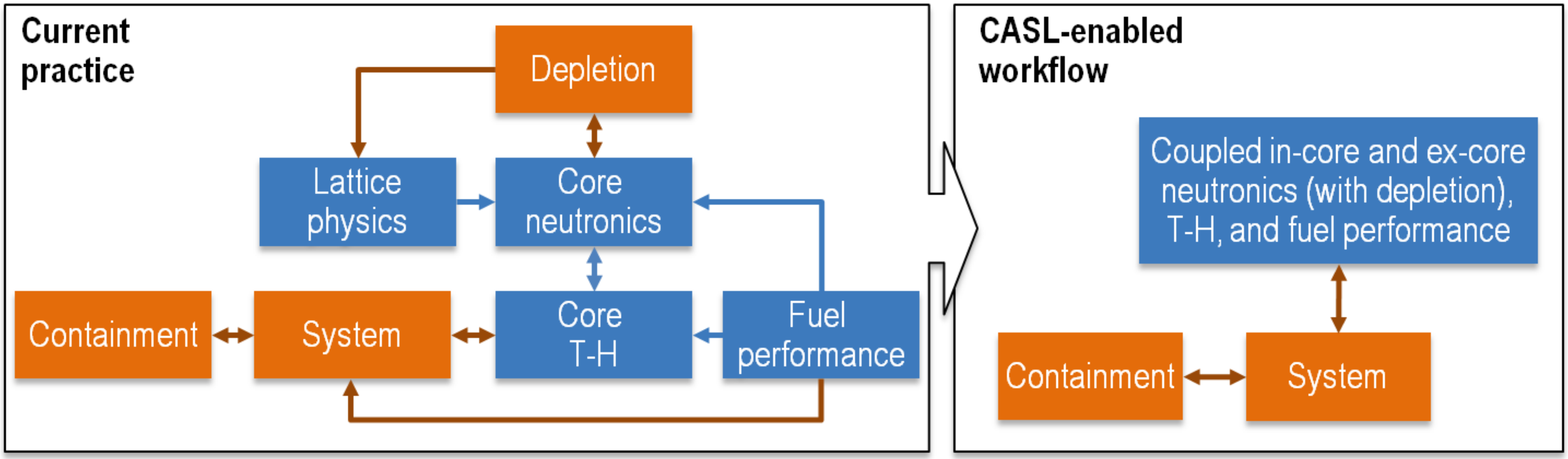
- Completed
  - RELAP5-3D given to CASL and placed in repository
  - Initial stand-alone integration of RELAP5-3D complete
  - RELAP5-3D build system converted to CMake
- Ongoing/future work
  - Complete inclusion of RELAP5-3D in CASL automated testing
  - Continue development of model evaluator
  - Define an appropriate coupled application for RELAP5-3D
  - Perform further LIME development as new physics codes are introduced and coupled

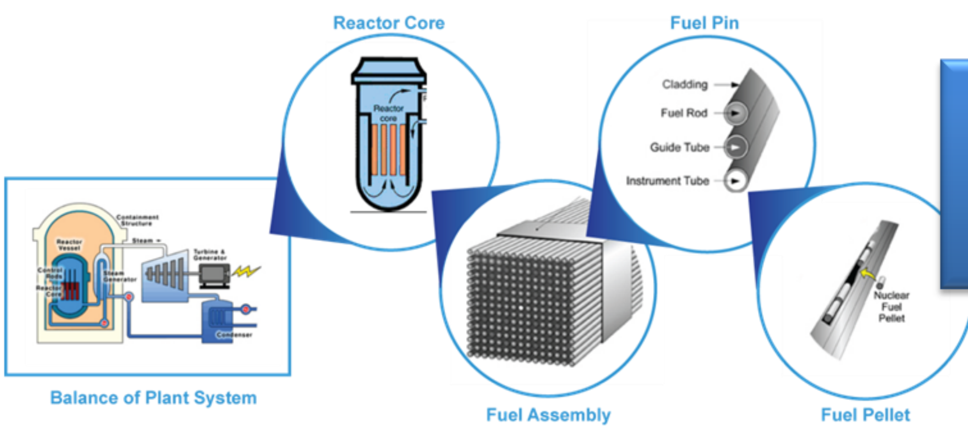
Questions?

www.casl.gov or info@casl.gov

Extra Slides

The CASL Virtual Reactor is at the heart of the plan and is the science and technology integrator





Suite of advanced yet usable M&S tools and methods, integrated within a common software infrastructure for predictive simulation of LWRs

Many coupling strategies are possible using LIME

- Choices available depend on what capabilities are in the physics codes being coupled
  - Restaurant analogy: Menu to choose from. You make choices, different items have different costs and value. You also might have dietary restrictions that preclude certain choices.
- Fixed point
  - Jacobi or Seidel options
  - Convergence based on “global residual” or “code by code”
- JFNK
  - Requires residuals, preconditioning recommended
- Alternate solvers for individual codes (NOX solver library in Trilinos)