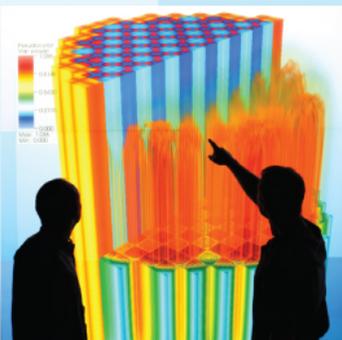


Power uprates
and plant life extension

CASL-U-2013-0164-000



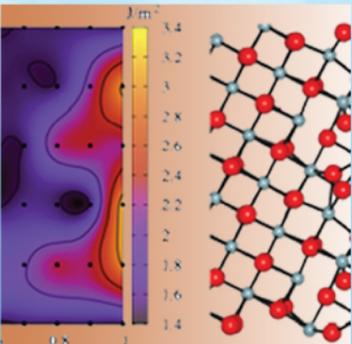
Engineering design
and analysis



Science-enabling
high performance
computing



Fundamental science



Plant operational data



Milestone L2:VRI.P7.01 Virtual Environment for Reactor Applications (VERA): Snapshot 3.1

07/24/2013

John A. Turner



U.S. DEPARTMENT OF
ENERGY

Nuclear Energy



Oak Ridge National Laboratory

in partnership with

- Electric Power Research Institute
- Idaho National Laboratory
- Los Alamos National Laboratory
- Massachusetts Institute of Technology
- North Carolina State University
- Sandia National Laboratories
- Tennessee Valley Authority
- University of Michigan
- Westinghouse Electric Company

and individual contributions from

- | | |
|-----------------------------|---------------------------------------|
| Anatech Corporation | Pacific Northwest National Laboratory |
| ASCOMP GmbH | Pennsylvania State University |
| CD-adapco, Inc | Rensselaer Polytechnic Institute |
| Core Physics, Inc. | Southern States Energy Board |
| City University of New York | Texas A&M University |
| Florida State University | University of Florida |
| Notre Dame University | University of Tennessee |
| Imperial College London | University of Wisconsin |

REVISION LOG

Revision	Date	Affected Sections	Revision Description
1	07/22/2013	All	Initial version
2	07/24/2013	All	Redid all table and figure referencing, moved sections around to match VERA 3.1 status, added figure for VERA 3.1, added evolution of coupled capability figure
3	07/28/2013	Many	Executive Summary, updated figures, coupling figures in several sections
4	07/31/2013	Many	added footer, updated Introduction, additional info in several component sections

EXECUTIVE SUMMARY

VERA, the Virtual Environment for Reactor Applications, consists of integrating and interfacing software together with a suite of physics components adapted and/or refactored to execute and, in some cases, simulate relevant physical phenomena in a coupled manner. VERA also includes the software development environment and computational infrastructure needed for these components to be effectively used.

Development of the Virtual Environment for Reactor Applications, VERA, is central to the CASL objective to “apply existing modeling and simulation capabilities and develop advanced capabilities to create a usable environment for predictive simulation of light water reactors.” This milestone marks snapshot 3.1 of VERA, and represents multiple improvements since the previous VERA snapshot.

Since the beginning of FY12 the emphasis has been on development of VERA’s “core simulator” functionality, which has included numerous enhancements to the neutronics capability (both transport and cross section processing), development of a common input strategy, integration of a subchannel thermal-hydraulic capability, and improvements to the VERA infrastructure components to enable analysis of challenge problem scenarios. Initial coupling began in FY12 and has continued in FY13, and VERA now includes a number of coupled capabilities in various stages of development:

- neutronics + CFD
 - Insilico + Drekar (not in active development, but successful demonstration)
- neutronics + subchannel thermal-hydraulics
 - COBRA-TF + Insilico (in use for analysis)
 - COBRA-TF + MPACT (in development)
- neutronics + subchannel thermal-hydraulics + fuel performance
 - COBRA-TF + Insilico + Peregrine (transitioning from development to testing)
- subchannel thermal-hydraulics + CRUD chemistry
 - COBRA-TF + MAMBA2D (in development and initial testing)

The components comprising the above capabilities, and the capabilities themselves, are described in more detail in this report and in other CASL milestone reports.

Naturally, the amount of software comprising VERA has grown as its capability has increased. VERA now consists of 5 infrastructure components and 11 physics components (5 of which provide coupled capabilities). There are currently over 5M lines of source code in over 30,000 files, housed in about 20 software repositories and organized in 240 packages. This complexity is managed by a system known as TriBITS which, by extending and working with widely used version control and build/test tools, maintains the environment in a coherent and scalable fashion, with continuous integration and over 700 nightly tests. In addition to the integrated VERA capabilities, the CASL software environment also includes standalone and coupled physics capabilities represented by baseline and prototype/demonstration codes.

CONTENTS

EXECUTIVE SUMMARY	iii
CONTENTS	iv
ACRONYMS	vi
LIST OF FIGURES	vii
LIST OF TABLES	ix
1 INTRODUCTION	2
2 SOFTWARE DEVELOPMENT TOOLS AND COMPUTATIONAL INFRASTRUCTURE	10
3 VERA INFRASTRUCTURE COMPONENTS.....	14
3.1 Lightweight Integrating Multiphysics Environment (LIME).....	14
3.2 VERA Common Input (VERAin)	17
3.3 Data Transfer Kit (DTK)	18
3.4 MOOSE.....	19
3.5 DAKOTA	19
4 VERA CORE SIMULATOR COMPONENTS	21
4.1 SCALE/XSProc	21
4.2 Insilico (Denovo).....	22
4.3 COBRA-TF.....	28
4.4 DAKOTA + COBRA-TF	29
4.5 COBRA-TF + Insilico.....	30
4.6 MPACT	33
4.7 Peregrine2D.....	35
4.8 Tiamat: COBRA-TF + Insilico + Peregrine2D.....	36
5 OTHER VERA COMPONENTS.....	39
5.1 Hydra-TH.....	39
5.2 MAMBA2D.....	40
5.3 COBRA-TF + MAMBA2D.....	41
5.4 RELAP5-3D	42
5.5 Drekar	43
5.6 Insilico-Drekar.....	44
6 VERA PHYSICS CAPABILITIES UNDER DEVELOPMENT.....	45
6.1 COBRA-TF + MPACT	45
7 VERA BASELINE COMPONENTS	47

7.1	ANC9.5	47
7.2	VIPRE-W	47
7.3	DAKOTA-VIPRE-W	47
7.4	ANC9.5-VIPREW (ANCLVIPRE)	48
7.5	ANC9.5-VIPREW-BOA	49
7.5.1	Multiphysics Coupling Diagram	50
7.5.2	Time advancement of the coupled multiphysics system	51
7.5.3	Comparison of selected results for a Watts Bar Cycle 1 calculation	55
8	VERA INITIAL AND DEMONSTRATION CAPABILITIES	58
8.1	DeCART	58
8.2	Star-CCM+	58
8.3	DeCART-Star-CCM+	59
8.4	Insilico-Star-CCM+	61
9	CONTRIBUTING STAFF	62
	Appendix A. VERA Input Format (ASCII)	2
	Appendix B. VERA Input Format (Parameters)	8
	Appendix C. DataTransferKit Reference	9
	Appendix D. Insilico Reference	10

ACRONYMS

AMA	Advanced Modeling Applications
CASL	Consortium for Advanced Simulation of Light Water Reactors
CILC	CRUD-induced localized corrosion
CIPS	CRUD-induced power shift
CFD	computational fluid dynamics
CRUD	corrosion-related unidentified deposits or Chalk River unidentified deposits
DOE	U.S. Department of Energy
DOE-NE	U.S. Department of Energy Office of Nuclear Energy
EIH	Energy Innovation Hub
EPRI	Electric Power Research Institute
FA	Focus Area
GTRF	grid-to-rod-fretting
HPC	high-performance computing
IC	Industry Council
INL	Idaho National Laboratory
LANL	Los Alamos National Laboratory
LWR	light water reactor
MIT	Massachusetts Institute of Technology
MPO	Materials Performance and Optimization
NCSU	North Carolina State University
NPP	nuclear power plant
NRC	Nuclear Regulatory Commission
NSSS	nuclear steam supply system
ORNL	Oak Ridge National Laboratory
PCI	pellet-cladding interaction
PWR	pressurized water reactor
RIA	reactivity insertion accident
RSICC	Radiation Safety Information Computational Center
RTM	Radiation Transport Methods
SNL	Sandia National Laboratories
T-H	thermal-hydraulics
THM	Thermal Hydraulics Methods
TVA	Tennessee Valley Authority
UM	University of Michigan
V&V	verification and validation
VERA	Virtual Environment for Reactor Applications
VOCC	Virtual Office, Community, and Computing
VRI	Virtual Reactor Integration
VUQ	Validation and Uncertainty Quantification
WEC	Westinghouse Electric Company

LIST OF FIGURES

Figure	Page
Figure 1.1: Desired functionality of VERA.	4
Figure 1.2: Components that comprise the VERA Core Simulator (VERA-CS).	4
Figure 1.3: Components of VERA 1.0.	5
Figure 1.4: Components of VERA 2.0.	5
Figure 1.5: Components of VERA 3.1.	6
Figure 1.6: Evolution of Coupled Capabilities in VERA.....	6
Figure 3.1: Key components of a simple generic application created using LIME.	15
Figure 3.2: A Representative Hierarchical Multiphysics problem.....	16
Figure 3.3: VERA Input Process.	18
Figure 3.4: Using TriKota to Bridge DAKOTA and LIME.....	20
Figure 4.1: Insilico has successfully completed AMA progression problems 1-6.	22
Figure 4.2: Visualization of reactor core power from a Denovo calculation.	23
Figure 4.3: Exnihilo package architecture and dependencies.	24
Figure 4.4: Old Exnihilo metadata design.	25
Figure 4.5: New Exnihilo metadata design.	26
Figure 4.6: 17x17 assembly automatically meshed using the Denovo neutronics package.	28
Figure 4.7: The coupled CTF+Insilico capability is being used for AMA progression problem 6.	30
Figure 4.8: Components comprising coupled capability for AMA progression problem 6.....	31
Figure 4.9: Components of the coupled Insilico-CTF application created to solve the example problem.....	31
Figure 4.10: Simplified flow chart illustrating the coupled code “Seidel” fixed point algorithm.	32
Figure 4.11: Graphical Output of the Fission Rate, Coolant Density, and Fuel Temperatures at 0 and 1300 ppm boron.	33
Figure 4.12: Status of MPACT w.r.t. AMA Progression Problems.....	35
Figure 4.13: Tiamat provides additional capability for AMA Progression Problem 6 and beyond.....	36
Figure 4.14: VERA Components comprising Tiamat.	37

Figure 4.15: Data transfers between physics components in Tiamat.....	37
Figure 4.16: Depiction of the MPI communication layers in Tiamat.	38
Figure 4.17: Surface plots of fission rate (from Insilico) and temperature in Peregrine. The plot on the right is scaled to show clad temperatures.....	38
Figure 4.18: Insilico averaged fuel temperature and fission rate.....	39
Figure 5.1: VERA components comprising coupled CTF+MAMBA2D capability.....	42
Figure 6.1: CTF+MPACT provides a pin-resolved capability for AMA progression problem 6.	45
Figure 6.2: VERA components used in an initial pin-resolved capability for AMA progression problem 6.....	46
Figure 7.1: Data transfer for DAKOTA-VIPRE-W.	47
Figure 7.2: Components used in ANC-VIPRE-W coupled baseline capability (ANCLVIPRE).....	48
Figure 7.3: Components used in ANC-VIPRE-W-BOA coupled baseline capability.....	49
Figure 7.4: LIME-based Code-Coupling Diagram for ANC9.5-VIPREW-BOA.	50
Figure 7.5: A physics-focused coupling diagram showing data dependencies and data exchange paths.	51
Figure 7.6: Data-flow diagram for a time-consistent multi-physics time advancement scheme based on a predictor-corrector method.	52
Figure 7.7: Data flow between the physics codes ANC-VIPRE-BOA for both the explicit predictor-corrector and the implicit Crank-Nicolson time-advancement methods.....	54
Figure 7.8: Comparison of boron mass predicted during Watts Bar Cycle 1.....	56
Figure 7.9: Comparison of Axial Offset predicted for Watts Bar Cycle 1.....	57
Figure 8.1: Components used in DeCART-Star-CCM+ capability.	59
Figure 8.2: Star-CCM+ (Left) and DeCART (Right) mesh for demonstration problem.....	60

LIST OF TABLES

Table.....	Page
Table 1.1: VERA Infrastructure Components.	8
Table 1.2: VERA Core Simulator Components.	8
Table 1.3: Other VERA Components.	8
Table 1.4: VERA Physics Components Under Development.	8
Table 1.5: VERA Baseline Components.	8
Table 1.6: Initial and Demonstration Capabilities.	9
Table 2.1: List of CASL Git Repositories.	11
Table 4.1: Comparison of AMA Benchmark 2 using VERA Common Input.	35
Table 8.1: Comparison of DeCART-Star-CCM+ serial and parallel solution and run times.	61

1 INTRODUCTION

This report summarizes the capabilities represented by snapshot 3.1 of the CASL Virtual Environment for Reactor Applications, VERA, and represents significant improvements and additional capabilities since the initial major VERA snapshot, version 1.0, completed for a March 31, 2011 milestone.

Figure 1.1 illustrates the general functionality needed in VERA. Much of the recent VERA development has focused on a collection of components within VERA that we have been referring to as the VERA “core simulator”. We recognize that although the term is commonly used in the nuclear industry, it can have multiple definitions. For CASL, this is neutronics, subchannel thermal-hydraulics, and fuel performance. For VERA, this is illustrated in Figure 1.2, and we often refer to this functionality collectively as VERA-CS. Note that it is not a single product or executable, and that the specific components and functionality comprising VERA-CS can and will evolve over time. For example, at some point CFD capabilities may become part of VERA-CS.

Figure 1.3 through Figure 1.5 illustrate the components of snapshots 1.0 (from March, 2011), 2.0 (from March, 2012), and 3.1 (the current snapshot), respectively. They include “legacy” tools (which provide a baseline capability), initial and demonstration capabilities (such as the LIME-coupled DeCART-Star-CCM+ capability) and advanced modeling and simulation tools being developed and incorporated within the VERA environment. In addition to the colors, which roughly represent components of similar type, darkness is used to connote the degree of integration with the environment. Note for example the difference between the light orange of the Insilico box in Figure 1.3 and the dark orange in Figure 1.4 and Figure 1.5. This is a simplified representation of a complex software system, and should not be interpreted to imply full coupling between components. However, VERA is evolving at a rapid pace, as evidenced by comparison between Figure 1.3 through Figure 1.5.

Efforts in FY13 have produced numerous enhancements to the neutronics capability (both transport and cross section processing), development of a common input strategy, integration of a subchannel thermal-hydraulic capability, and improvements to the VERA infrastructure components to enable analysis of challenge problem scenarios. Initial coupling of advanced components began in FY12 and has continued in FY13, and VERA now includes a number of coupled capabilities in various stages of development:

- neutronics + CFD
 - Insilico + Drekar (not in active development, but successful demonstration) – Section 5.6
- neutronics + subchannel thermal-hydraulics
 - COBRA-TF + Insilico (in use for analysis) – Section 4.5
 - COBRA-TF + MPACT (in development) – Section 6.1
- neutronics + subchannel thermal-hydraulics + fuel performance
 - COBRA-TF + Insilico + Peregrine (transitioning from development to testing) – Section 4.8
- subchannel thermal-hydraulics + CRUD chemistry
 - COBRA-TF + MAMBA2D (in development and initial testing) – Section 5.3

Section 2 provides a description of the software development environment and computational infrastructure that currently supports the development of VERA.

Section 3 through Section 8 describe the computer software and physics codes that comprise VERA. Sub-sections describe each of the major components, including some illustrative results on relevant problems when

available. Important codes not yet incorporated into the official VERA build and test environment but obtained by CASL and currently available in the VRI repository are also described.

Section 9 lists the staff members from various CASL institutions who contributed to the completion of this milestone.

We note that many details of the work described in this summary report are also documented on the CASL VRI Kanban site:

https://casl-dev.ornl.gov/trac/casl_vri_kanban

where a series of tickets can be found detailing the Epics, Stories and Tasks that were completed, reviewed, and closed as a part of this milestone.

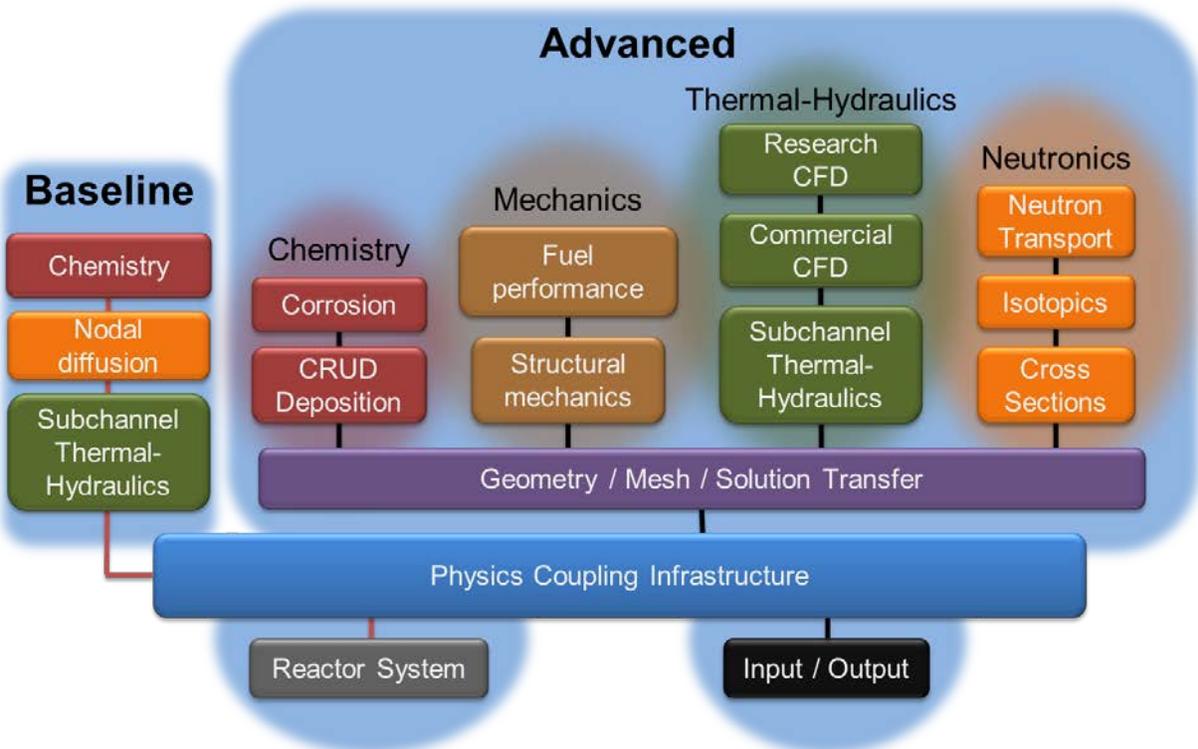


Figure 1.1: Desired functionality of VERA.

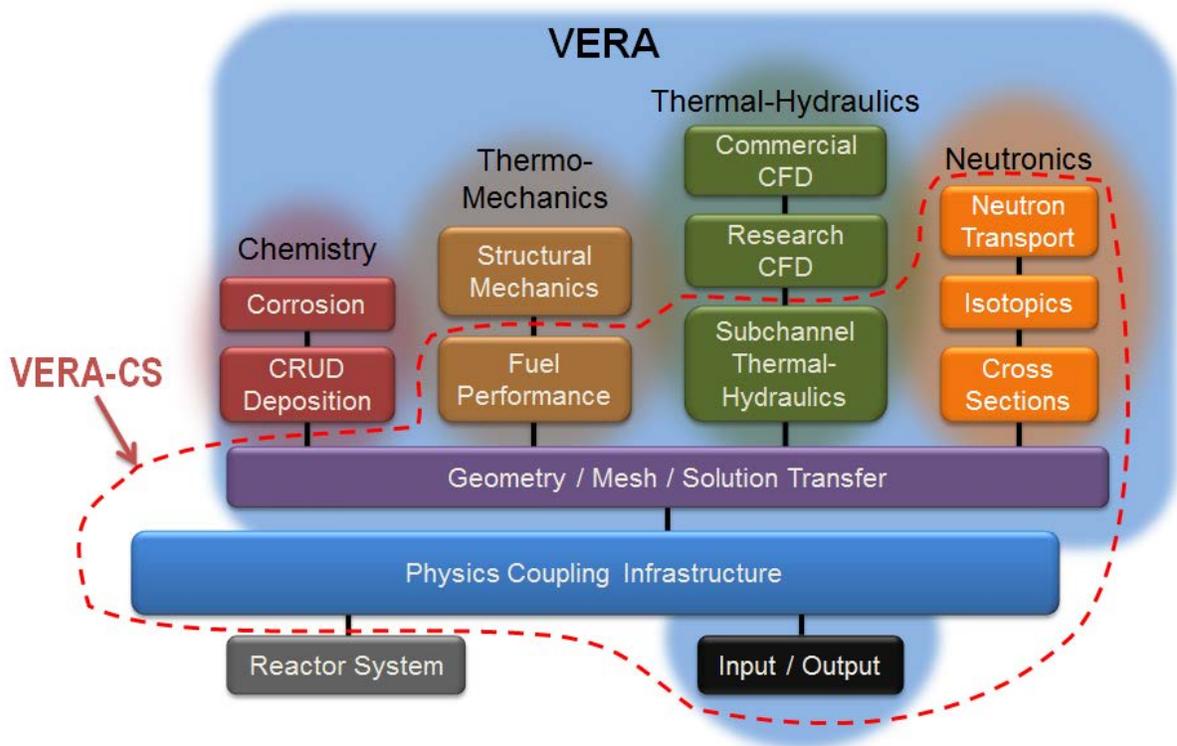


Figure 1.2: Components that comprise the VERA Core Simulator (VERA-CS).

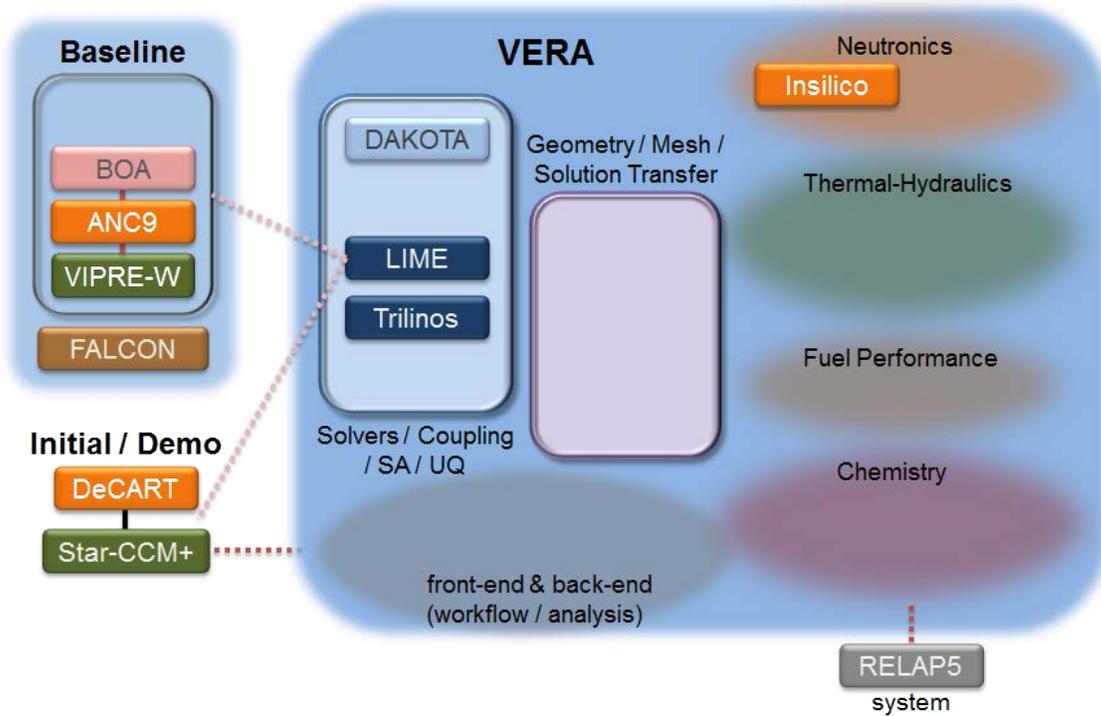


Figure 1.3: Components of VERA 1.0.

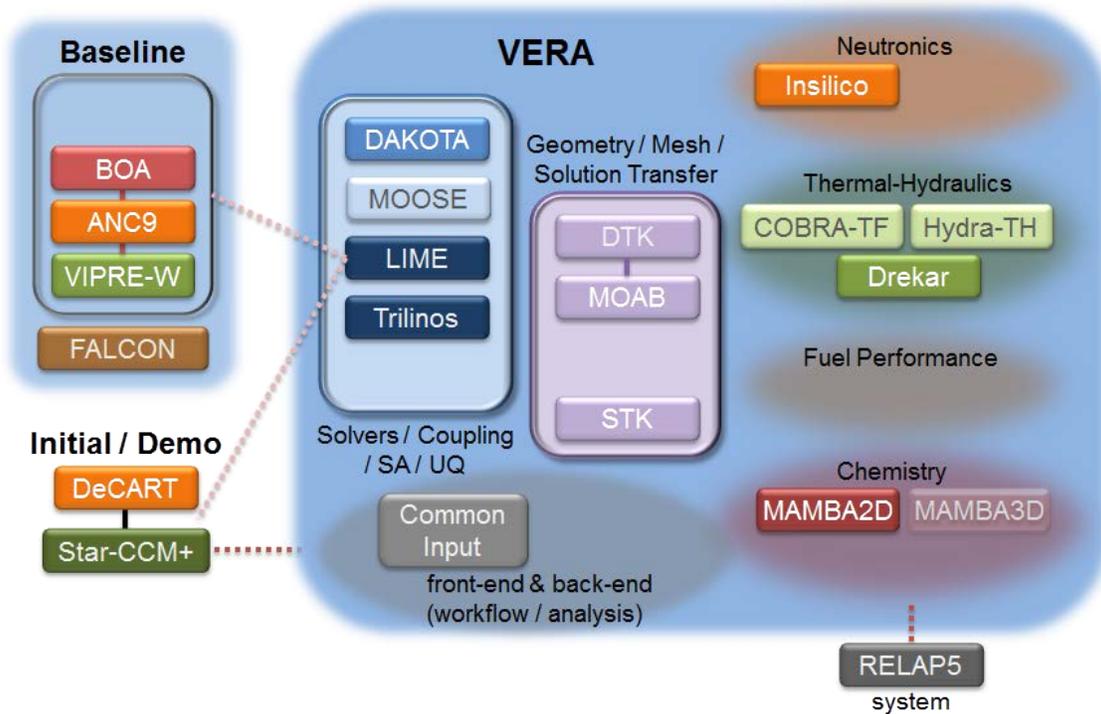


Figure 1.4: Components of VERA 2.0.

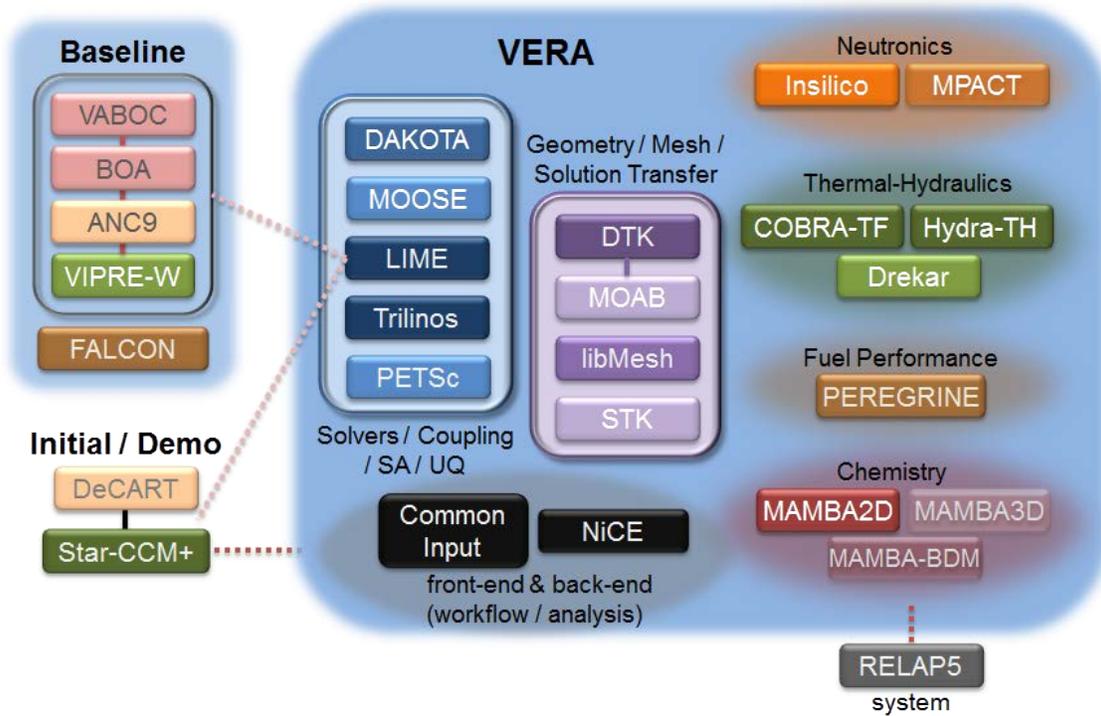


Figure 1.5: Components of VERA 3.1.

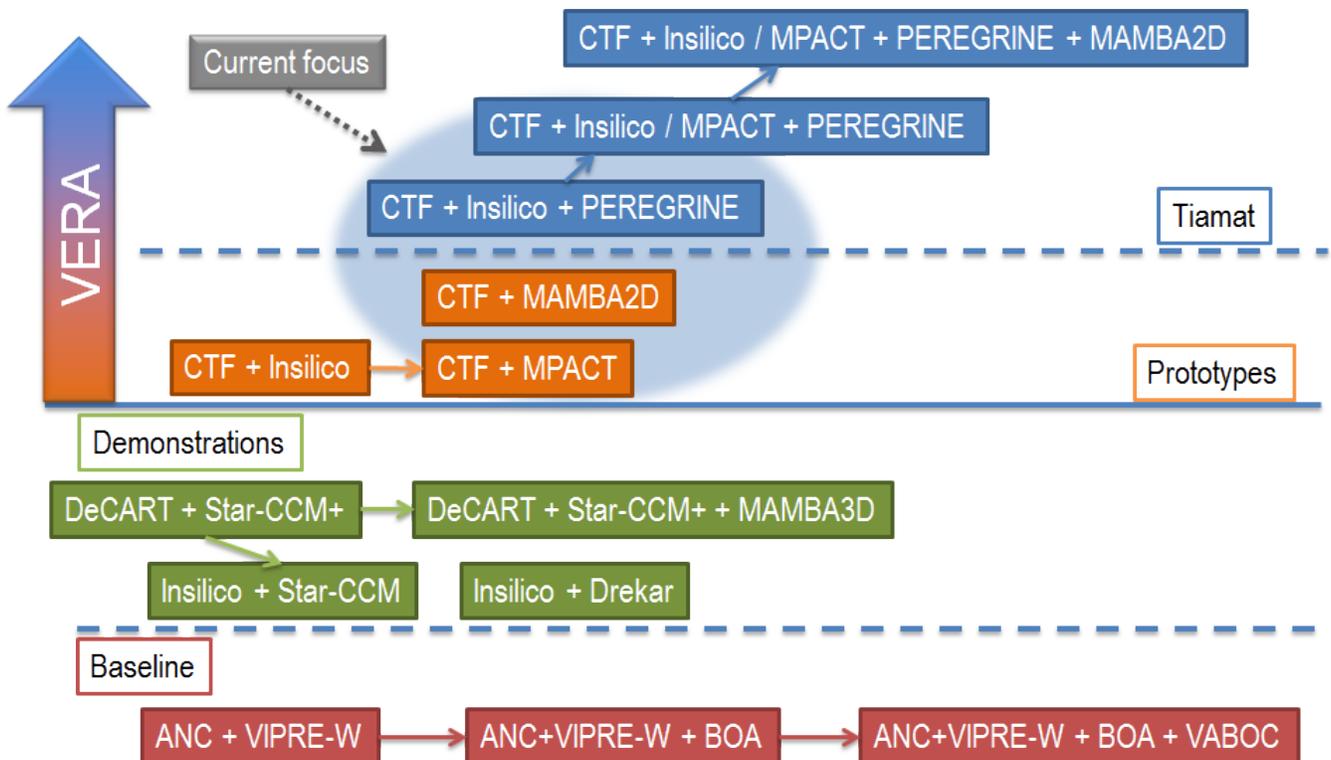


Figure 1.6: Evolution of Coupled Capabilities in VERA.

Table 1.1 through Table 1.6 give an overview of the activity, integration status, and approximate assessment of capability for each of the VERA components for this snapshot:

- Activity – The approximate level of VRI activity devoted to this component / capability for this snapshot. Darker indicates more intense activity. Note that this does not indicate development activities occurring within other CASL Focus Areas (e.g. THM-funded development of Hydra-TH, RTM-funded development of MPACT, etc.).
- Integration – Approximate assessment of the status of integration of this component into VERA as of this snapshot. Yellow indicates some level of integration has been completed (e.g. source code has been obtained, code has been ported to development platforms, etc.). Green indicates that the majority of the integration tasks have been completed (e.g. under nightly build/test).
- Capability – Approximate assessment of capability provided through VERA, as measured against desired capability to be accessible through VERA. No indication of a component’s capability outside VERA is intended. Yellow indicates some level of capability (possibly significant). Green indicates that the capability provided is approaching the level required.

Table 1.1: VERA Infrastructure Components.

Component	Description	Activity	Integration	Capability
LIME	Coupling			
Common Input	Usability			
DTK	Solution transfer			
MOOSE	Framework			
DAKOTA	SA / UQ			

Table 1.2: VERA Core Simulator Components.

Component	Description	Activity	Integration	Capability
SCALE / XSPROC	Cross sections			
Insilico	Pin-homogenized neutronics			
COBRA-TF (CTF)	Subchannel Thermal-Hydraulics (T-H)			
DAKOTA + CTF	Subchannel T-H with SA / UQ			
CTF + Insilico	Coupled neutronics and T-H			
MPACT	Pin-resolved neutronics			
Peregrine2D	Fuel performance			
Tiamat	CTF + Insilico + Peregrine2D			

Table 1.3: Other VERA Components.

Component	Description	Activity	Integration	Capability
RELAP5-3D	Reactor System			
Drekar	CFD			
Insilico + Drekar	Coupled neutronics and CFD			
Hydra-TH	CFD			
MAMBA2D	CRUD			
CTF + MAMBA2D	CRUD			

Table 1.4: VERA Physics Components Under Development.

Component	Description	Activity	Integration	Capability
CTF + MPACT	Coupled pin-resolved neutronics and T-H			

Table 1.5: VERA Baseline Components.

Component	Description	Activity	Integration	Capability
ANC 9.5	Nodal neutronics			
VIPRE-W	Subchannel Thermal-Hydraulics (T-H)			
DAKOTA + VIPRE-W	Subchannel T-H with SA / UQ			
ANC9.5 + VIPRE-W	Coupled nodal neutronics and T-H			
ANC9.5 + VIPRE-W + BOA	Coupled nodal neutronics, T-H, and CRUD			

Table 1.6: Initial and Demonstration Capabilities.

Component	Description	Activity	Integration	Capability
DeCART	Pin-resolved neutronics			
Star-CCM+	Commercial CFD			
DeCART + Star-CCM+	Coupled neutronics and CFD			
Insilico + Star-CCM+	Coupled neutronics and CFD			

2 SOFTWARE DEVELOPMENT TOOLS AND COMPUTATIONAL INFRASTRUCTURE

The VERA suite of software is the set of software designed to provide a set of capabilities for use by both internal (AMA) and external CASL customers. Thus there are significant software quality issues that must be considered with respect to VERA. As a result, the VRI Infrastructure Team (who maintains and supports the official VERA development environment) has defined a 'VERA Lean/Agile Development Model ' (see https://casl-dev.ornl.gov/wiki/index.php/VRI_Development_Model) that includes the following key elements:

- All functionality of VERA is defined and protected by automated tests. These tests provide real-time evidence for the correct functioning of VERA software. These tests are maintained with 100% success at all times.
- The official VERA sources and tests are to be maintained in working order through a set of pre-push continuous integration (CI) tests, post-push CI tests (i.e. CTest/CDash [2] system), and Nightly testing. (All supported by the TriBITS system [1,2].)
- New functionality is developed incrementally, while constantly maintaining existing functionality. Functionality is developed in small increments where constant refactoring is used to maintain the current functioning of the software.

The software described in this section is felt to either satisfy, or very nearly satisfy these requirements. Other physics components reside in the VRI repository and are being developed, but that do not yet meet the above criteria.

The software development environment includes all of the components necessary to reliably build, test and run CASL software components. Challenges include:

- The need to support difference classes of users with different needs, i.e., analysts running codes as well as developers building/testing codes.
- The need to support multiple, distinct platforms.
- The need to support the integration and maintenance of codes from multiple sources and multiple development teams.

The VERA software development environment consists of the following:

- A standardized set of compilers, tools and third-party libraries, driven by the requirements of the CASL software components.
- Tools present on the CASL computational platforms for configuring the operating environment to use/develop particular CASL software components.
- The Git revision control system [3] and software repository.
- Build and test infrastructure and methodology, utilizing the TriBITS system [4,5] based on CMake [1]/CTest/CDash [2] using standard practices developed and employed for the Trilinos project.

Each software component in VERA is accessible from the CASL-VRI git software repository to those who have the proper access permissions. Fine-grained access controls allow development and deployment of proprietary codes. This system provides

- automated revision control and tracking,
- enforcement of IP protection, and
- accessibility of the software for authorized CASL staff.

Table 2.1 lists the repositories currently associated with VERA.

Table 2.1: List of CASL Git Repositories.

Repository Name	Repository	Checkout Location
VERA TPLs	casl-dev:/git-root/casl_tpls.svn	
VERA	casl-dev:/git-root/VERA	VERA
Trilinos (CASL devs)	casl-dev:/git-root/Trilinos	VERA/Trilinos
... Trilinos devs	<i>software.sandia.gov:/space/git/Trilinos</i>	
LIME (CASL devs)	casl-dev:/git-root/LIMEExt	VERA / LIMEExt
... LIME devs	<i>software.sandia.gov:/space/git/LIMEExt</i>	
Data Transfer Kit	casl-dev:/git-root/DataTransferKit	VERA / DataTransferKit
VERA Input Support	casl-dev:/git-root/VERAInExt	VERA / VERAINEx
VRI PSS drivers	casl-dev:/git-root/casl_vripss	VERA / PSSDriversExt
Westinghouse codes	casl-dev:/git-root/casl_rave	VERA / CASLRAVE
BOA	casl-dev:/git-root/casl_boa	VERA / CASLBOA
DeCART (CASL devs)	casl-dev:/git-root/casl_decart	VERA / DeCARTEExt
... DeCART devs	<i>arc-05.engin.umich.edu:/git-root/DeCART/decart-v2.git</i>	
MOOSE	casl-dev:/git-root/casl_moose	VERA / MOOSEExt
Peregrine	casl-dev:/git-root/casl_peregrine	VERA / PeregrineExt
MPACT (CASL devs)	casl-dev:/git-root/casl_mpact	VERA / MPACTExt
... MPACT devs	<i>arc-05.engin.umich.edu:/git-root/MPACT/MPACT.git</i>	
RELAP5-3D	asl-dev:/git-root/casl_relap5	VERA / RELAP5Ext
StarCCM	casl-dev:/git-root/StarCCMClient	VERA / StarCCMExt
Scale (Hg repo)	casl-dev:/git-root/casl_scale	VERA / Scale
Exnihilo (CASL devs)	casl-dev:/git-root/Exnihilo	VERA / Exnihilo
... denovo devs	<i>angmar.ornl.gov:/data/git/Exnihilo.git</i>	
Nemesis (CASL devs)	casl-dev:/git-root/nemesis	VERA / Nemesis
... denovo devs	<i>angmar.ornl.gov:/data/git/nemesis.git</i>	
MAMBA	casl-dev:/git-root/casl_mamba	VERA / MAMBAExt
MAMBA-BDM	casl-dev:/git-root/casl_mamba_bdm	VERA / MAMBA-BDM
COBRA-TF	casl-dev:/git-root/casl_cobra	VERA / COBRAExt
Panzer (CASL devs)	casl-dev:/git-root/Panzer	VERA / Panzer
... Panzer devs	<i>software.sandia.gov:/space/git/Panzer</i>	
Hydra-TH	casl-dev:/git-root/hydrath	VERA / hydrath

An important aspect of the VERA software development environment is the automated continuous-integration build and test system based on the TriBITS system. Whenever a software change is made, either to the code itself or to any other code that it depends on, the software is immediately tested for successful completion of a suite of code-specific test problems. A more rigorous set of tests are performed each night to ensure that software capabilities have not been inadvertently compromised during the development process. Without such a system to continuously build and test integrated software, experience has shown that these capabilities can erode due to changes in interfaces and functionality, portability problems, build system changes, etc. Software that is continuously built and tested delivers fewer defects and allows daily local releases of software (which is critical to meeting CASL goals).

Additional online documentation is now available at the VRI Kanban Trac site [6]. This can be accessed through the following links that appear at the bottom of the main page.

- CASL Repository Access Instructions
- CASL VRI Software Development Environment
- CASL Software Quality Assurance
- VERA Software Release History
- Quality Program Description

The CASL computational platforms currently include:

- Fissile four: Four 32-core, shared-memory machines (u233, u235, pu239, pu241). These provide CPU and memory resources to allow simultaneous development, testing and small-scale computation by multiple users. One machine (pu241) is reserved for nightly and continuous-integration testing of all CASL software components; the other three are for development of CASL software components.
- Boris and Natasha: Two 64-core shared-memory machines recently added to enhance development and testing.
- Fission: A 12,512-core Appro cluster at INL. Approximately 2000 cores have been allocated for CASL development and production use. Components of VERA have been ported to Fission and are being used by AMA staff.
- Jaguar (deprecated): ~300,000-core Cray XT5 system at ORNL. Has now been upgraded to the Cray XK7 system known as Titan.
- Titan [8]: ~300,000-core Cray XK7 system with NVIDIA K20X-accelerated nodes. Titan is currently undergoing acceptance and stability testing and not yet available to users.
- Frost (deprecated): This is a 1024-core cluster for running medium- to large-scale computations. Originally serving as the main CASL development platform, Frost exists now as a deployment target. Currently, there are restrictions as to what software can be run on Frost (and how), specifically proprietary codes such as those owned by Westinghouse.
- Darter: JICS/NICS Cray.

References for Section 2:

1. <http://www.cmake.org/>
2. <http://www.cdash.org/>

3. git, the fast distributed version control system, <http://git-scm.com/about>
4. Bartlett, Roscoe, Michael Heroux, and Jim Willenbring. TriBITS Lifecycle Model Version 1.0: A Lean/Agile Software Lifecycle Model for Research-based Computational Science and Engineering and Applied Mathematical Software. SAND2012-0561. Sandia National Laboratories. February 2012
5. <http://code.google.com/p/tribits/>
6. https://casl-dev.ornl.gov/trac/casl_vri_kanban
7. https://casl-dev.ornl.gov/trac/casl_vri_kanban/wiki/CaslRepoAccess
8. <http://www.olcf.ornl.gov/titan/>

3 VERA INFRASTRUCTURE COMPONENTS

The VRI Physics Simulation Suite consists of integrating and interfacing software together with a suite of physics codes that have been adapted and/or refactored to run and, in some cases, be coupled together within the VERA environment. A summary description of each of these that are associated with this release is given in the subsections that follow.

3.1 Lightweight Integrating Multiphysics Environment (LIME)

LIME is a small software package for helping to create multiphysics simulation codes. The name is an acronym denoting “Lightweight Integrating Multiphysics Environment for coupling codes.” LIME is especially useful when separate computer codes already exist to solve different parts of a multiphysics problem. LIME provides high-level software (written in C++), a well-defined but flexible approach, and interfaces to enable the assembly of multiple physics codes into a single coupled multiphysics simulation code.

For CASL, LIME has been an important software tool for coupling physics components. For example, LIME is used to create the coupled multiphysics simulation capabilities of ANC9.5-VIPREW-BOA and of DeCART-Star-CCM+. In addition, many of the other VERA physics codes have been “wrapped” under LIME, a first step in preparation for coupling with other physics codes.

Version 1.0 of LIME has been released under a BSD-type open source license, and both a users guide[1] and a theory manual [2] have been completed and are now available. Although LIME 1.0 contains the basic functionality required for many uses, it is not a fully mature tool, and improvements to LIME are being pursued as needed to address CASL requirements. Enhancements needed to achieve early milestones are being incorporated into updates to the current version of LIME (LIME 1.x). More significant changes addressing longer-term improvements/enhancements identified as important to CASL are expected to be reflected in a 2.x version of LIME.

Because every multiphysics code-coupling scenario is unique in some aspects, some amount of customized software must be written. In addition, depending on the design of the original physics codes, some degree of refactoring of the components being coupled may also be required. In other words, LIME is not “plug and play” in the way that this phrase is normally interpreted. However, the work required to write the customized software and to potentially re-factor the physics codes can be very small relative to the alternative of creating an entirely new multi-physics code from scratch. This is particularly relevant when decades of R&D have been invested in computer codes, which is true for many being used by industry today.

Figure 3.1 illustrates the key software components needed for creating a new multi-physics application within VERA using LIME 1.0. For each physics code being coupled a customized “Model Evaluator” is written, and for each new multiphysics application a small but unique “Multi-physics Driver” is needed. In this figure physics codes A, B, or C, generically represent any three CASL codes that might be coupled together (for example ANC9, VIPRE-W and BOA), but the number of codes shown is simply illustrative.

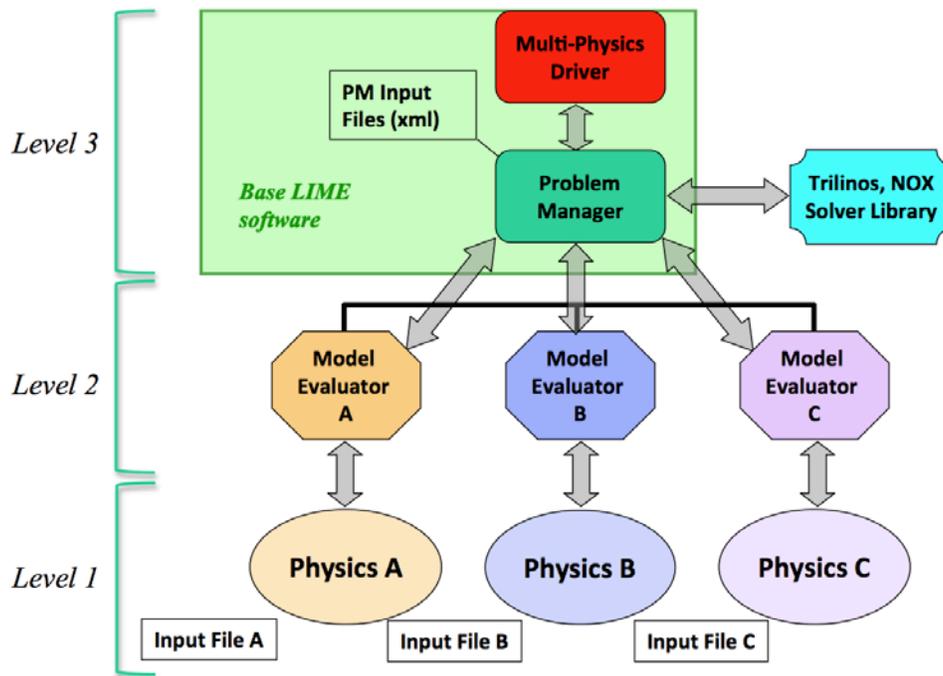


Figure 3.1: Key components of a simple generic application created using LIME.

LIME is built as an extension of Trilinos [3], and leverages several Trilinos packages and libraries, such as NOX [4]. To solve the coupled non-linear systems presented by coupled physics codes, LIME currently supports both fixed-point (Jacobi and Seidel) and Jacobian-Free Newton Krylov (JFNK) methods (with a predictor option for transients). When using JFNK, LIME attempts to leverage as many of the Trilinos/NOX options that the underlying physics codes can support.

LIME is compatible with serial or parallel codes written in any standard language and can couple transient and/or steady-state problems. The range of physics codes being brought together within VERA span all of these variations.

Hierarchical Solves: LIME provides the ability to structure a multiphysics problem as a collection of hierarchies. An example of such a problem is illustrated in Figure 3.2 and is representative of what might be needed when modeling a depletion transient in a reactor. Here we see that the Problem Manager directly sees only two physics-code Model Evaluators (labeled “Steady State Solve” and “DeCART-Deplete,” respectively). However, the Steady State Solve actually couples two different physics codes; one modeling fluid flow (denoted Star CCM+) and another Neutron Transport (denoted DeCART-Transport).

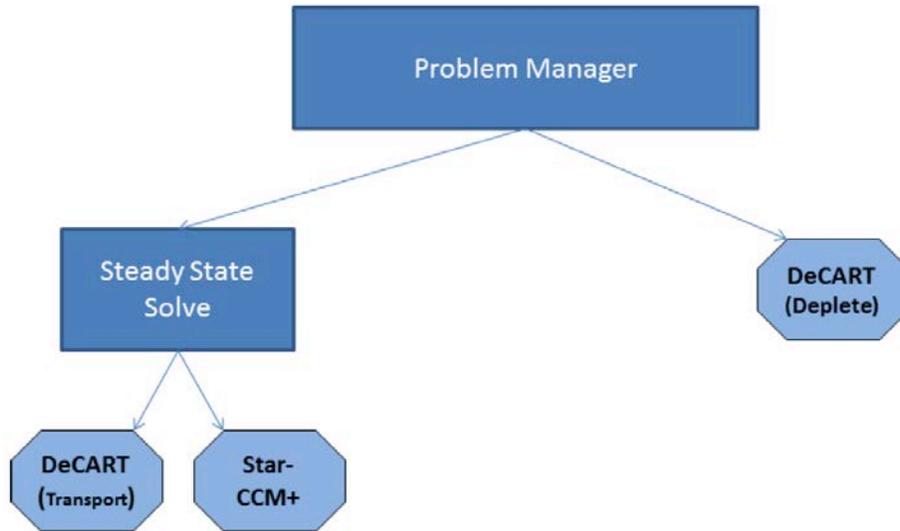


Figure 3.2: A Representative Hierarchical Multiphysics problem.

Enhancements to LIME enable this type of problem to be more easily set-up and solved. A simple test problem has been created and placed into the LIME testing suite and the approach for setting this type of problem up has been documented and added to an update working version of the users manual.

Fixed Point Acceleration: Several methods exist to potentially improve convergence robustness and rate of convergence of standard fixed-point algorithms. They are generally known as mixing methods and are based on using previous iterates when computing the next solution update, e.g. for linear problems convergence akin to GMRES can be obtained.

One such method, called Anderson Acceleration, has now been implemented for testing purposes in LIME 1.x and a simple example problem added to the LIME testing suite. This capability has not yet been exposed via a user interface or tested on large-scale nonlinear problems, but it does show much promise. The particular method used in LIME is taken from and documented in [5], and the LIME 1.x users manual is being updated to reflect this added capability. LIME's implementation is currently limited to serial execution (it will require a parallel QR algorithm to work in parallel) and produces the expected GMRES convergence behavior for a linear problem. Of note is that the linear problem in the unit test does not converge using a traditional (non-accelerated) fixed-point method, thus illustrating one important advantage of this type of approach.

References for Section 3.1:

1. R. Schmidt, N. Belcourt, R. Hooper and R. Pawlowski, An Introduction to LIME 1.0 and its Use in Coupling Codes for Multiphysics Simulations, SAND2011-8524, Sandia National Laboratories, Albuquerque, New Mexico, November 2011.
2. R. Pawlowski, R. A. Bartlett, N. Belcourt, R. Hooper, and R. Schmidt. A theory manual for multi-physics code coupling in LIME. Sandia Technical Report SAND2011-2195, Sandia National Laboratories, March 2011.
3. Sandia National Laboratories. "The Trilinos Project". <http://www.trilinos.sandia.gov>.

4. Sandia National Laboratories. “NOX and LOCA, Object-Oriented Nonlinear Solver and Continuation Packages”. <http://www.trilinos.sandia.gov/packages/nox>.
5. Peng Ni, Anderson Acceleration of Fixed-point Iteration with Application to Electronic Structure Computations, PhD Dissertation, Mathematical Sciences, Worcester Polytechnic Institute, Nov. 13, 2009.

3.2 VERA Common Input (VERAin)

The VERA Common Input (VERAin) is a single common input used to drive all of the physics codes in the VERA Core Simulator (VERA-CS). Early in the development of the core simulator, it was recognized that it would be unreasonable to require users to generate input decks for each of the individual physics codes. This is especially true if the core simulator allows multiple codes to solve each physics problem (i.e. multiple subchannel codes, multiple neutronics solvers). In addition to the ease-of-use aspects, it is critical in multiphysics applications that all of the different code systems have consistent input. Having a single common input simplifies the user experience and helps ensure that all of the physics applications are solving a consistent geometry.

The common input is based on a single ASCII input file. The input file uses a free-form input format that is based on keyword inputs. The format of the input file was designed by engineers with broad experience with current industry core design tools, so the format of the input file will be easy for industry users to understand. The ASCII input file provides several advantages to the users:

- Allows users to easily transfer input and output between different computer systems.
- Allows users ability to easily edit the file on remote computers.
- Provides a format that users can readily read and understand.
- ASCII input files are an approved archive format recognized by the NRC (ASCII, PDF, or TIFF).
- Allows users to “diff” input files on a variety of remote computers
- Allows users to archive inputs in standard source code repositories and/or directories with read-only permissions.

The input file contains a description of the physical reactor geometry, including: fuel assemblies, removable poison assemblies, control rods, non-fuel structures, detectors, baffle, etc. The input file also contains a description of the current reactor statepoint including: power, flow, depletion, search options, etc.

Figure 3.3 illustrates the overall design of the Common Input capability VRI has designed and implemented for VERA. Note that the definitive input file is in the XML format and can be generated in multiple ways (text, script, or GUI). In order to translate the user input to input needed for the individual components, a multistep process is used. First, an input parser reads the text input file and converts it into an XML file. Some physics components, such as Insilico and MPACT, can read the XML file directly using readily-available XML libraries. Other components, such as CTF and Peregrine, require an intermediate step that converts the XML file into the native code input. This process allows the common input file to be used for existing physics codes where we do not want to (or cannot) make extensive modifications to the input.

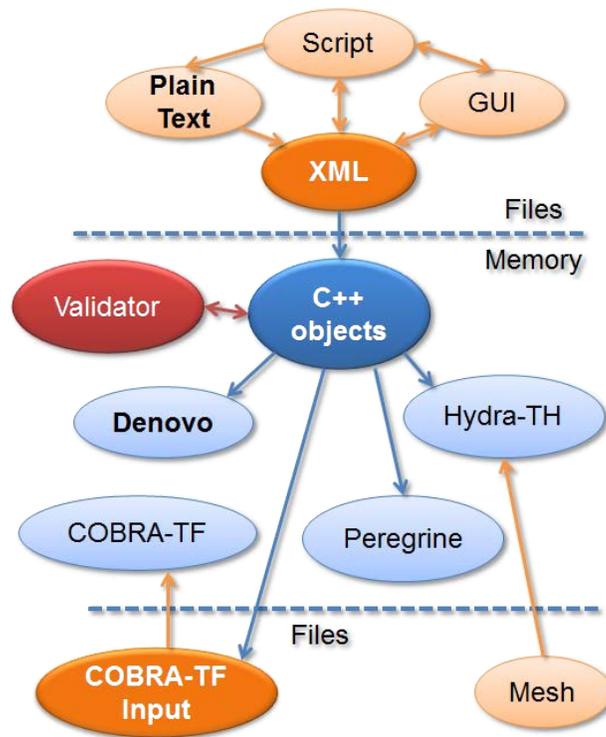


Figure 3.3: VERA Input Process.

A sample input deck can be found in Appendix A. Extensive documentation of the text input capabilities, as well as that of the XML representation, can be found in Appendix B. That information is in general kept up-to-date on the CASLpedia web site.

As of VERA Snapshot 3.1, four components are using the common VERA input capability:

- Insilico
- COBRA-TF
- MPACT
- Peregrine2D (initial capability)

plus three coupled capabilities:

- CTF + Insilico
- CTF + Insilico + Peregrine2D
- CTF + MAMBA2D

3.3 Data Transfer Kit (DTK)

In order to provide an infrastructure for data transfer operations between CASL physics codes as an alternative to custom schemes developed for each combination of physics, work has been done over the last two periods of development to provide a set of tools for parallel data transfer. This software, designated as the DataTransferKit (DTK), provides a set of interfaces and tools that application developers can use to aid in the parallel transfer of data between physics codes. To date, much of the development has been focused on the design of these

interfaces that provide the link to each physics code participating in coupling as well integrating this software into the VERA development environment.

As documented in the VERA 2.0 snapshot report, significant enhancements to DTK were implemented in response to recommendations documented in appendix C of the report for CASL milestone L3.VRI.PSS.P4.02 [1]. This included redesign of some DTK internals as well as implementation of the rendezvous algorithm for improved capability and enhanced error handling [2].

References for Section 3.3:

1. Level 3 Milestone Deliverable Report VRI.PSS.P4.02
2. Data Transfer Kit Error Handling Policy, Revision 0 (Attached to TRAC ticket #2542)

3.4 MOOSE

A snapshot of the Multiphysics Object-Oriented Simulation Environment (MOOSE) resides in the CASL repository and has been ported to the CASL development platforms. MOOSE [1] is required for the Peregrine fuel performance capability. See CASL milestone report L2.VRI.P7.02 (CASL-I-2013-0165-000) for details on Peregrine (and MOOSE) integration into VERA [2].

References for Section 3.4:

1. D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie´. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. Nucl. Eng. Design, 239, p. 1768–1778, 2009.
2. R. Pawlowski, J. Turner, S. Palmtag, and R. Montgomery, “Initial Demonstration of Peregrine in VERA-CS,” L2.VRI.P7.02, CASL-I-2013-0165-000, July 31, 2013.

3.5 DAKOTA

DAKOTA provides a flexible interface between analysis codes and iterative systems analysis methods and contains a host of specialized algorithms and tools for optimization, uncertainty quantification, and sensitivity/variance analysis. In CASL, DAKOTA primarily supports the goals and objectives of the Validation and Uncertainty Quantification (VUQ) focus area, though it is used by foundational and integrating focus areas as well. VERA integration and deployment of DAKOTA makes its capabilities readily available to all CASL partners and focus areas. Extensive documentation and help mechanisms for DAKOTA are available at the DAKOTA website <http://dakota.sandia.gov/>.

DAKOTA was integrated into VERA as part of the first major snapshot (VERA 1.0) and no significant changes with respect to its functionality in VERA have been implemented since that time. A number of studies have been conducted with DAKOTA and VERA simulation tools, notably

1. with VIPRE-W in a March 2011 L2 Milestone VUQ.P2.03 (a.k.a. VUQ.Y1.03)
2. with VIPRE-W and BOA in a December 2011 L1 Milestone CASL.P4.01, and
3. by CASL industry partners for both calibration and uncertainty quantification.

These all demonstrate filesystem-based “black-box” coupling of DAKOTA and simulation tools, one important means of integration in VERA.

Another important VRI focus area task has more tightly incorporated DAKOTA into VERA, developing interfaces between DAKOTA and LIME-based multi-physics applications. The current interface is based on TriKota, a Trilinos package that adapts DAKOTA’s parameter/response API to a Trilinos ModelEvaluator. Thus DAKOTA and LIME can be compiled together and run to exercise sensitivity analysis, uncertainty quantification, optimization, or calibration of LIME-coupled multi-physics problems. This is illustrated in Figure 3.4.

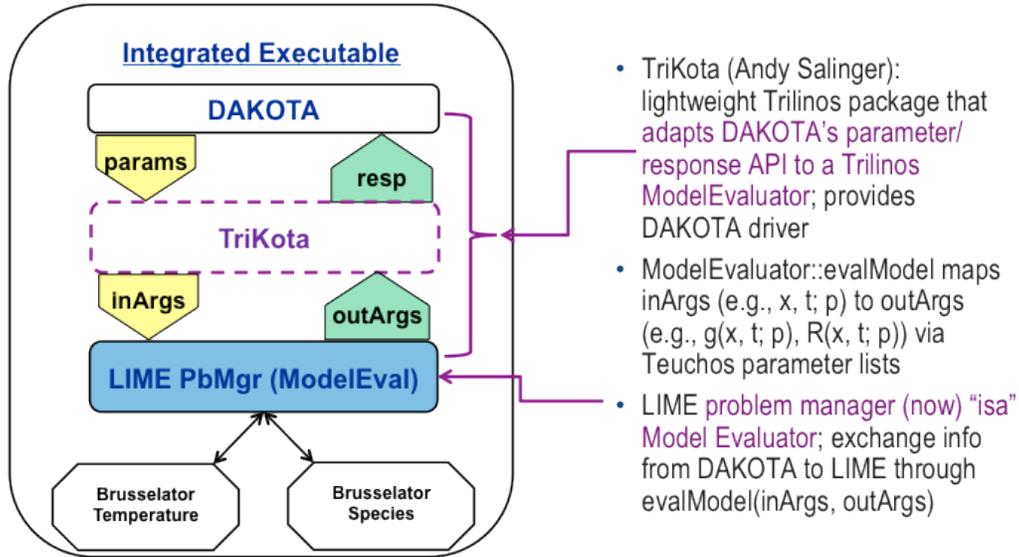


Figure 3.4: Using TriKota to Bridge DAKOTA and LIME.

4 VERA CORE SIMULATOR COMPONENTS

4.1 SCALE/XSProc

The XSProc module is based on techniques in the SCALE code system [1] for the generation of problem-dependent multigroup cross-section data and has been incorporated into VERA for use in neutronics calculations. XSProc performs resonance self-shielding with full range Bondarenko factors using either the narrow resonance approximation or the intermediate resonance approximation. For uniform fuel lattices, Dancoff factors are automatically generated from the user-input geometry and material descriptions. XSProc also allows user-input Dancoff factors to treat non-uniform lattice effects. The fine energy-group structure of the resonance self-shielding calculation can optionally be collapsed to a coarse group structure through a one-dimensional (1D) transport calculation internal to XSProc.

The SCALE 6.1 238-group ENDF/B-VII.0 neutron cross-section data library is available for production calculations, and an 8-group library is available strictly for testing purposes. Both libraries contain data for 417 nuclides and 19 thermal-scattering moderators. Thermal scattering data for most nuclides are available at five temperatures, where other nuclides provide data for as many as 10 temperatures. Full-range Bondarenko factors are provided for resonance self-shielding calculations.

The XSProc module provides the following capabilities:

- Temperature interpolation,
- Problem-dependent resonance self-shielding,
- Macroscopic mixing of multigroup cross-section data,
- Energy collapse of cross-section data,
- Serialize/deserialize unit cells and cross-section data,
- Dancoff factor calculation, and
- User input Dancoff factors.

The analytical methods of XSProc are based on methods implemented in SCALE 6.1, which have been substantially refactored for use in VERA.

- Bondarenko calculations for resonance self-shielding: BONAMI is a module of the SCALE system (SCALE User Manual Section F1), which is used to perform Bondarenko calculations for resonance self-shielding. Cross sections and Bondarenko factor data are input from cross-section data library and an updated cross-section data library is generated. A wide variety of options are provided for different lattices and cell geometries through the use of Dancoff approximations. A novel interpolation scheme is used which avoids many of the problems of the widely employed Lagrangian schemes. Where cross-sections are requested at temperature other than those available on the cross-section data library, linear interpolation is performed.
- Material processing: The techniques of the SCALE Material Information Processor (SCALE User Manual Section M7) are included in XSProc to convert user input material data into atom number densities needed for cross-section processing; convert user-input of fuel lattices, region-wise, and infinite medium into a form suitable for cross-section processing; calculate Dancoff factors for uniform lattices; allow

user-input Dancoff factors for non-uniform lattices; and generate an appropriate spatial mesh for 1D transport calculations.

- One-dimensional transport for flux-weighted cross-section data: The XSDRNPM transport solver (SCALE User Manual Section F3) provides flux-weighting of cross-section data and generates group collapsed and cell-homogenized data for subsequent multi-dimensional transport calculations. XSDRNPM is a discrete-ordinates code that solves the 1D Boltzmann equation in slab, cylindrical, or spherical coordinates. A variety of calculation types are available, including fixed source, eigenvalue, or “search” calculations.
- Generation of macroscopic cross-section data: The Bondarenko method provides resonance self-shielded for microscopic cross-section data, but macroscopic cross-section data is needed for transport calculations. XSPROC provides macroscopic mixing of cross-section data using updated techniques that are similar to those of the ICE module of SCALE (SCALE User Manual Section F8).

References for Section 4.1:

1. SCALE: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, ORNL/TM-2005/39, Version 6.1, Oak Ridge National Laboratory, Oak Ridge, Tennessee, June 2011. Available from Radiation Safety Information Computational Center at Oak Ridge National Laboratory as CCC-785.

4.2 Insilico (Denovo)

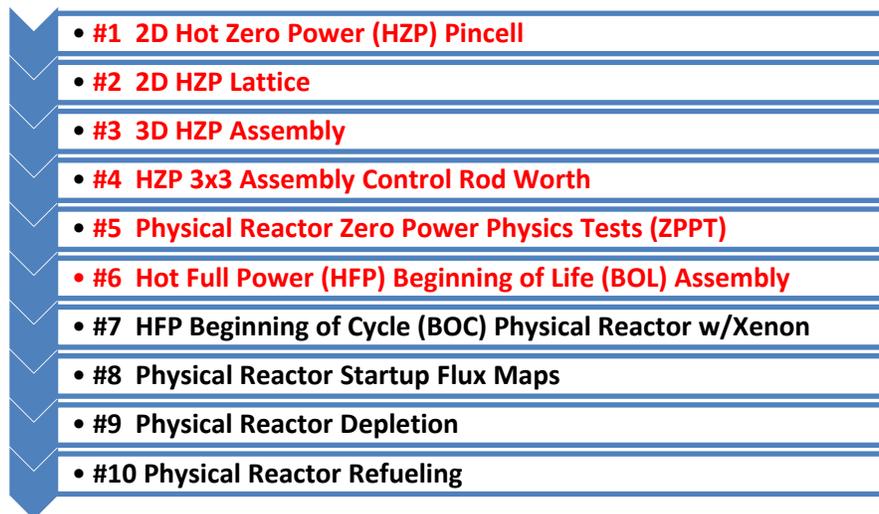


Figure 4.1: Insilico has successfully completed AMA progression problems 1-6.

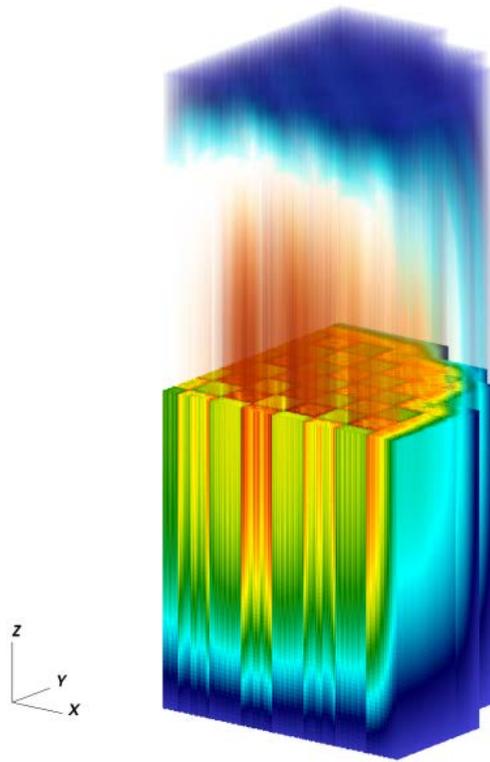


Figure 4.2: Visualization of reactor core power from a Denovo calculation.

A new neutronics package has been developed for the VERA Core Simulator (VERA-CS) [1]. This 2/3D core simulator uses SCALE/XSProc (Section 4.1) to generate cross-sections and Denovo to perform the transport calculations. The entire package is now referred to as Exnihilo. The Exnihilo code package is a set of components (sub-packages) that enables the construction of parallel deterministic and Monte Carlo transport applications. Exnihilo is based on a package architecture model such that each package provides well-defined capabilities, as shown in Figure 4.3.

Exnihilo currently contains five packages. These are

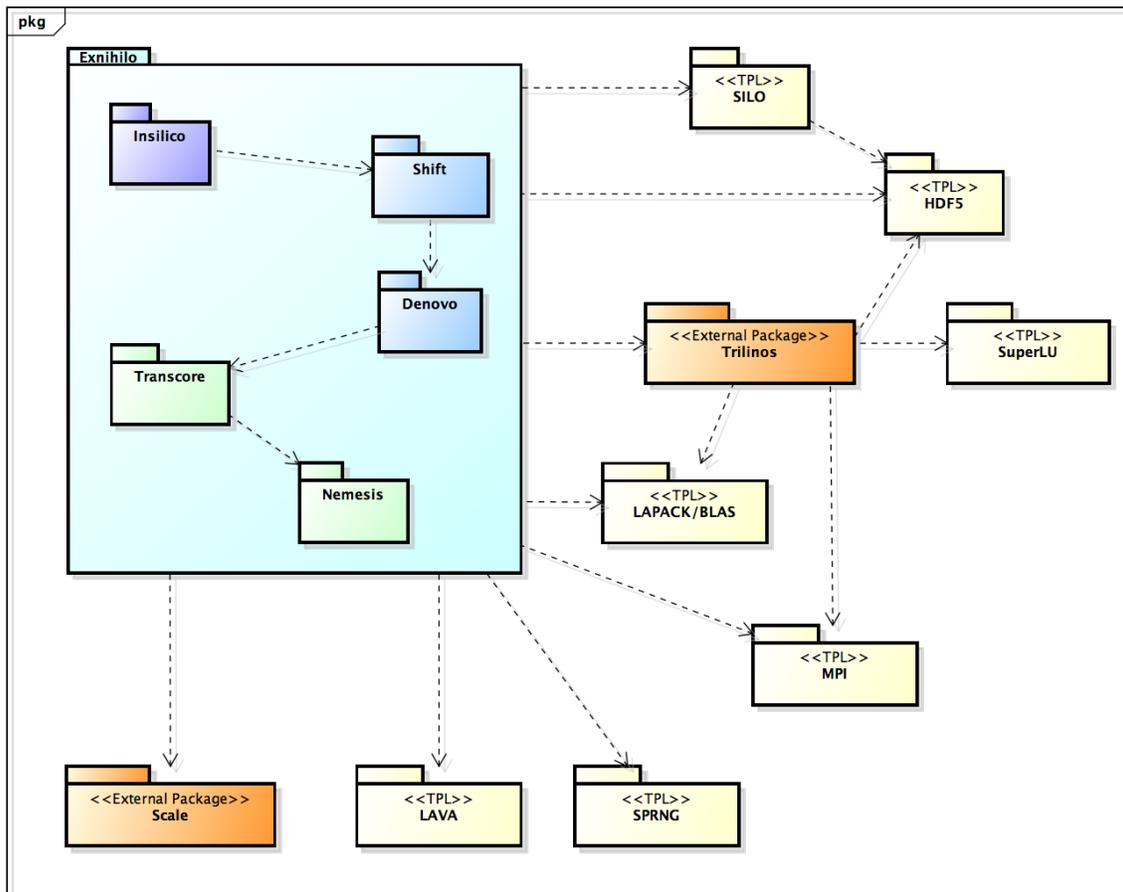
- **Nemesis:** testing, infrastructure, Design-by-Contract, and parallel communication (MPI) resources
- **Transcore:** database, quadrature, field, and other general components that are useful for building multiphysics applications involving transport
- **Denovo:** deterministic transport solvers and front-ends for fixed-source and eigenvalue problem for regular-grid S_N and SP_N and 2D MOC on combinatorial geometries.
- **Shift:** Monte Carlo and hybrid Monte Carlo fixed-source and eigenvalue solvers.
- **Insilico:** Neutronics front-end that couples Denovo and/or Shift with cross-section processing, depletion, and TH (thermo-hydraulics) feedback.

Exnihilo is built using a continuous-integration model that emphasizes test-driven design and software verification. It uses the TriBITS (Trilinos) CMAKE-based build system and Git for configuration management. More details can be found by examining the Doxygen-produced documentation (Appendix D. Insilico Reference).

Denovo is a three-dimensional radiation transport code under development at Oak Ridge National Laboratory. It uses the well-established S_N discretization and solves the transport equation on Cartesian grids using the Koch-Baker-Alcouffe wavefront parallel algorithm. Denovo uses a multi-level parallel decomposition over space-angle and energy, and reasonable scaling has been demonstrated using 200,000 processors on the Jaguar XT5 at OLCF.

Denovo uses the Trilinos library to implement state-of-the-art numerical solvers, including Krylov multigroup solvers and Arnoldi eigenvalue solvers, in addition to the standard fixed-point iteration methods such as DSA-accelerated source iteration for within-group iterations in a Gauss-Seidel multigroup outer iteration and power iteration for eigenvalue problems.

Pin-homogenized quarter core simulations have been performed with Denovo, and show excellent results, with 9 pcm difference from a large Denovo reference calculation in 23.7 minutes on 9600 cores. A more detailed description of development in Denovo is documented in [2].



powered by Astah

Figure 4.3: Exnihilo package architecture and dependencies.

In order to support Benchmark Problem 5, which requires simulating a full-core pressurized-water reactor with control rods, burnable poisons, detectors and other components, a refactoring of the metadata system in Exnihilo was required. In Figure 4.4, we have a schematic of the original metadata system in Exnihilo. User input (in the form of an XML file) was fed into the Metadata Builder. The Metadata Builder would then create assembly metadata, where each assembly was a collection of arrays of pincells. After the metadata builder was

finished, the metadata would pass to the Assembly Builder. The Assembly Builder would construct spacer grids by squeezing additional arrays into the array stack for each assembly.

Initially, this methodology worked fine. However, once control rods assemblies, control rod banks, burnable poison inserts, detectors, axial boundaries for output edits, and other constructs were considered, it was discovered that this methodology was too complex. The reason is because the Metadata Builder was tasked with constructing the core metadata before the entire core has been defined. It was then up to the Assembly Builder to “fix up” the metadata before passing it to the automatic meshing system (not shown) and Insilico.

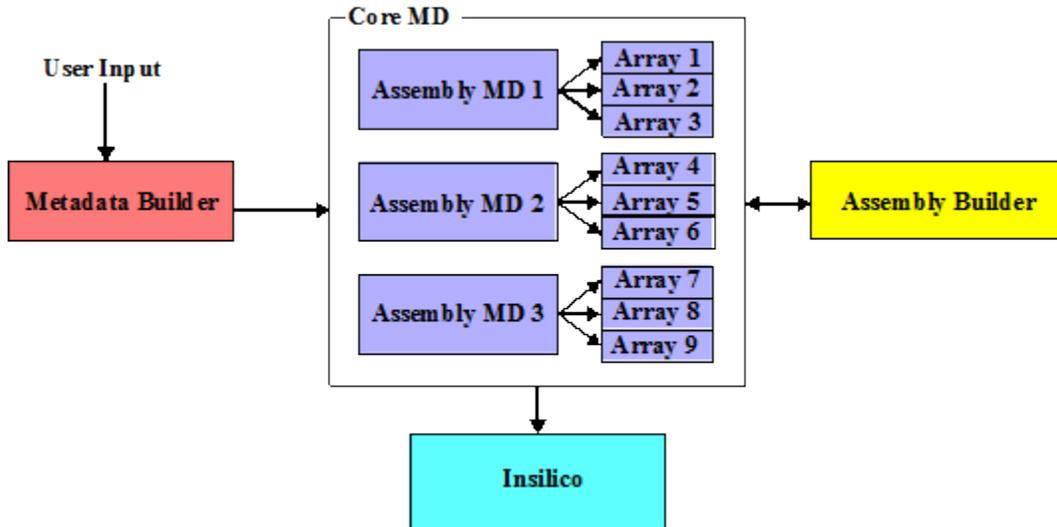


Figure 4.4: Old Exnihilo metadata design.

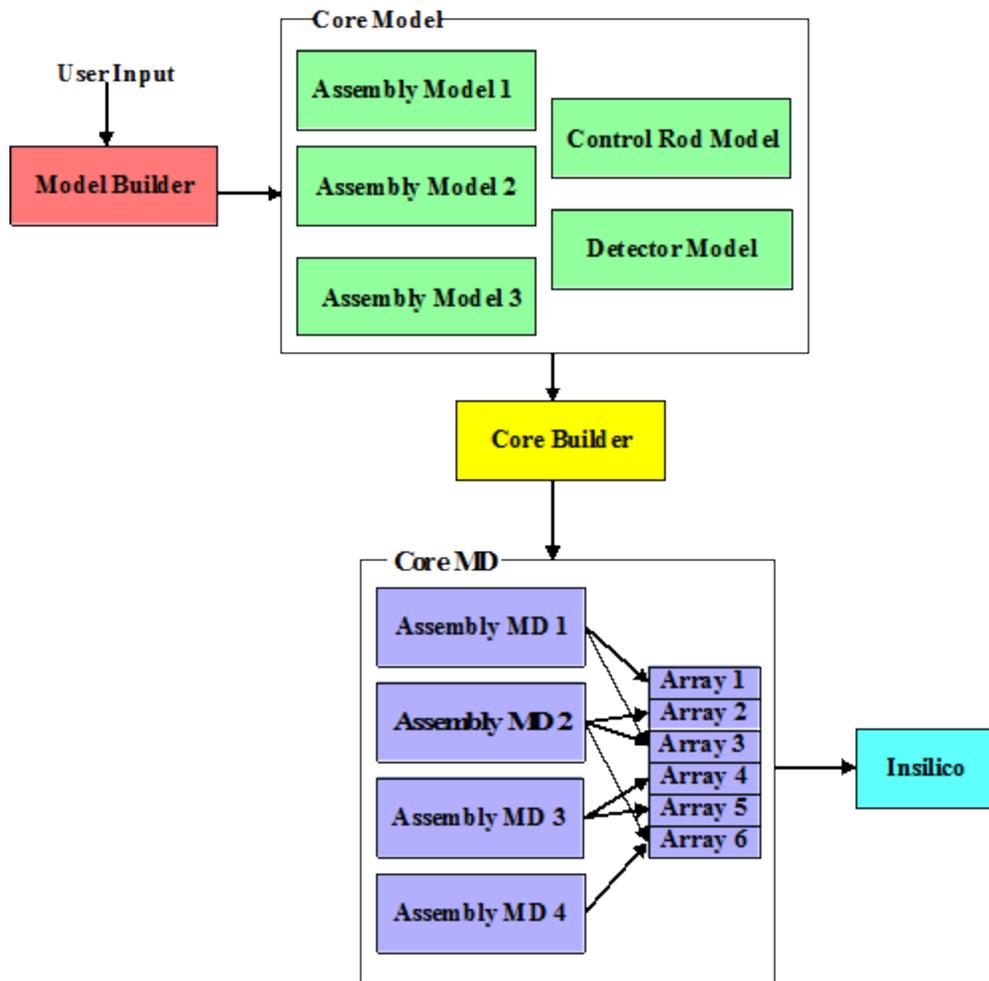


Figure 4.5: New Exnihilo metadata design.

The new metadata system, illustrated in Figure 4.5, adds a model component. Rather than using the input to generate metadata directly, the input is used to construct a complete model of the reactor core. Once the model has been completely defined, it is passed to the Core Builder, where the complete metadata infrastructure can be constructed. It is now easy to add multiple control rod banks, inserts, detectors, and arbitrary axial planes, since these are all specified before the metadata is constructed. The new metadata has the additional benefit of eliminating duplicate pincells and arrays across assemblies, which can reduce the memory footprint and computational expense when generating cross section data.

Additionally, the metadata is built from the top-down rather than bottom-up, as in the old system. This allows the Core Builder to recognize when arrays (or even individual pincells) are shared across assemblies. As a result, it is guaranteed that the minimum set of pincells (or arrays, in the case of ESSM), are passed to the cross section builder, thus minimizing the computational cost of the cross section calculations and the memory footprint of the cross section database.

In summary, the steps performed by VERA/Insilico are:

1. Process a reactor engineering specification converted into XML format
2. Convert input specification into arguments for XSPROC

3. Generate a geometric representation of the reactor needed by XSProc
4. Build and partition a Denovo discrete mesh representation of the reactor geometry
5. Broadcast XSProc geometry to each processor domain
6. Run XSProc to generate macroscopic cross sections
7. Map the cross sections to Denovo computational mesh cells
8. Run Denovo SN to calculate scalar fluxes
9. Integrate scalar fluxes with fission reaction rate data to calculate axial power distributions in each pin cell
10. Do parallel output

An example of the automated meshing is shown in Figure 4.6. These steps have been integrated into the Insilico neutronics package that is included in VERA.

The mesh is automatically mapped to the physical pin-cells that have cross sections generated by XSProc. All parallel communication is handled using the Denovo parallel framework.

Denovo/XSProc contains over 350 individual unit tests to verify the software implementation. The package has been run on AMA Test Problems 1-5, as well as forming the basis for the coupled capability used for AMA progression problem 6, as described in Section 4.5.

Recently, the Simplified Legendre (SP_N) solver was used to successfully complete an L1 milestone, as documented in the report for L1.CASL.P7.01 [3].

The entire package is parallel; problem 3 was performed on TITANDev on 1024 cores. Automated problem specification and meshing required less than 1s of wall-clock time.

VERA-CS is tested nightly on Linux-x86_64 platforms. We have also verified the software on MacOSX (10.6, 64-bit) and the Cray XK6. Denovo builds using GNU, Intel, or PGI compiler suites. However, XSProc requires Intel or GNU ($\geq 4.6.1$). Executables have also been built for the INL Fission platform for AMA use.

References for Section 4.2:

1. “VERA Technical Requirements by Component” (CASL-2011-0074-000-CI)
2. Evans, T.M., Davidson, G.G., and Jarrell, J.J., “Denovo-3_0_0,” RNSD-TN-11-003.
3. J. Gehin, A. Godfrey, F. Franceschini, T. Evans, B. Collins, S. Hamilton, “Operational Reactor Model Demonstration with VERA: Watts Bar Unit 1 Cycle 1 Zero Power Physics Tests”, L1.CASL.P7.01, CASL-U-2013-0105-000, June 7, 2013.

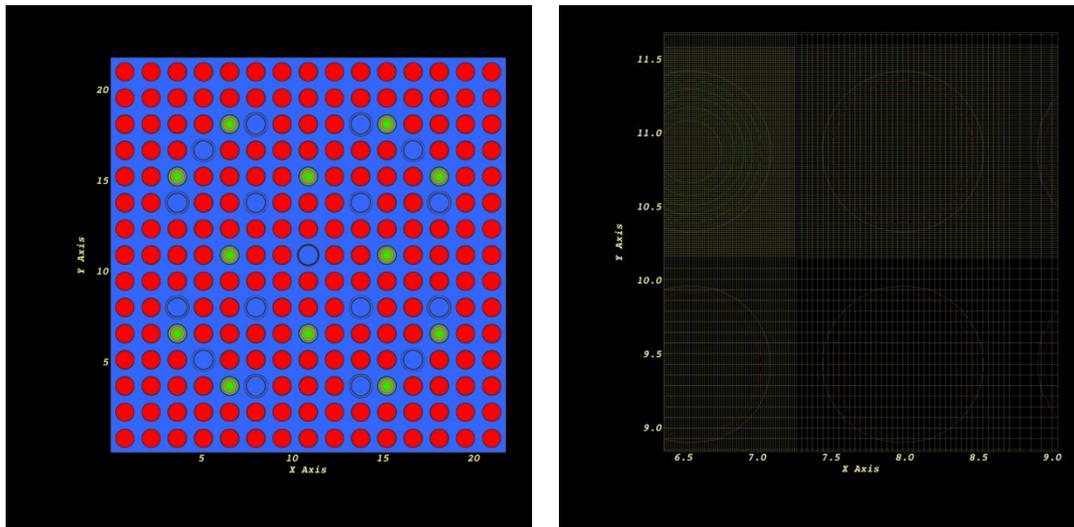


Figure 4.6: 17x17 assembly automatically meshed using the Denovo neutronics package.

4.3 COBRA-TF

COBRA-TF is a thermal-hydraulic simulation code based on sub-channel approximations commonly used for Light Water Reactor (LWR) core analysis. It uses a two-fluid, three-field (i.e. liquid film, liquid drops, and vapor) modeling approach. Both sub-channel and 3D Cartesian forms of nine conservation equations are available for LWR modeling. The code was originally developed by Pacific Northwest Laboratory in 1980 and has been used and modified by several institutions over the last few decades.

CASL initially obtained a public domain version of COBRA-TF held by the Reactor Dynamics and Fuel Management Group (RDFMG) at the Pennsylvania State University (PSU). This version of COBRA [1, 2, 3] – here denoted CTF – includes a variety of improvements made at PSU. CTF is capable of capturing two-phase flow transient behavior and includes a robust combination of LWR modeling tools.

In CTF, the conservation equations for each of the three fields and for heat transfer from and within the solid structure in contact with the fluid are solved using a semi-implicit, finite-difference numerical technique on an Eulerian mesh, where time intervals are assumed to be long enough to smooth out the random fluctuations in the multiphase flow, but short enough to preserve any gross flow unsteadiness. The fluid volume is partitioned into a number of computational cells. The equations are solved using a staggered difference scheme. The phase velocities are obtained at the cell faces, while the state variables - such as pressure, density, enthalpy, and void fraction - are obtained at the cell center. The momentum equations are solved on staggered cells that are centered on the scalar mesh face.

CTF is developed for use with either 3D Cartesian or sub-channel coordinates and features flexible nodalization for both the thermal-hydraulic and the heat-transfer solution. This flexibility allows a fully 3D treatment in geometries amenable to description in a Cartesian coordinate system and the use of the sub-channel approximation for faster calculations when the flow is principally in one direction.

The code is able to handle both hot wall and normal flow regimes maps and it is capable of calculating reverse flow, counter flow, and cross-flow situations.

Improvements to COBRA associated with Penn State CTF version include the following:

- The source code has been transitioned to the FORTRAN 90 standard.
- Attention has been given to improving code error checking and the input deck has been converted from fixed to free format.
- Krylov solver numerical techniques have been implemented to enhance computational efficiency.
- Improvements have been made to the turbulent mixing and direct heating models and code quality assurance testing has been performed using an extensive LWR validation and verification matrix.
- Code documentation (including mathematical modeling, programming, and user manuals) have been generated.
- The two-fluid formulation, generally used in thermal-hydraulic codes, separates the conservation equations of mass, energy, and momentum to vapor and liquid. CTF extends this treatment to three fields: vapor, continuous liquid and entrained liquid droplets, which results in a set of nine time-averaged conservation equations.
- New physical models have been implemented, including
 - a point kinetics model;
 - a second order implicit boron tracking algorithm [4];
 - a horizontal flow regime;

In addition to numerous additional performance improvements, VERA is now able to generate single-assembly CTF input decks from the common input described in Section 3.2. Multiple-assembly input is under development.

References for Section 4.3:

1. M. Avramova, D. Cuervo, K. Ivanov, "Improvements and Applications of COBRA-TF for Stand-Alone and Coupled LWR Safety Analyses", Proc. of PHYSOR-2006 Conference, Vancouver, Canada (2006).
2. M. Avramova, K. Ivanov, L. Hochreiter, "Analysis of Steady State and Transient Void Distribution Predictions for Phase I of the OECD/NRC BFBT Benchmark using CTF/NEM," NURETH-12 Conference, Pittsburgh, Pennsylvania, U.S.A. September 30-October 4 (2007).
3. D. Cuervo, M. Avramova, K. Ivanov, R. Miro, "Evaluation and Enhancement of COBRA-TF Efficiency for LWR Calculations," Annals of Nuclear Energy, Volume 33, pp. 837-847 (2006).
4. O.E. Ozdemir, M. Avramova, K. Sato, "Boron Tracking Model Implementation in COBRA-TF: Analytical and Sensitivity Analysis", 2010 ANS Winter Meeting, Las Vegas, Nevada, USA, November 2010, Vol. 103, pp. 1008-1009.

4.4 DAKOTA + COBRA-TF

The integration of DAKOTA described in Section 3.5 was initially used to perform sensitivity analysis using the Westinghouse subchannel thermal-hydraulic code VIPRE-W, as described in Section 7.3. This infrastructure was leveraged to provide an equivalent capability using the non-proprietary COBRA-TF subchannel thermal-hydraulic code.

4.5 COBRA-TF + Insilico

The Denovo-based neutronics capability Insilico has been coupled with CTF in support of AMA problem 6, Hot Full-Power Beginning of Life Assembly (Figure 4.7). Figure 4.8 depicts the VERA components used for this capability, Figure 4.9 illustrates the coupling between the components, Figure 4.10 provides an overview of the solution algorithm, and Figure 4.11 provides initial results for a 17x17 assembly, showing fission rate, coolant density, and fuel temperature for 0 ppm and 1300 ppm boron. The axial shift in power shape due to T/H feedback is readily apparent.

For further details of this capability and more extensive simulation results, see the report for milestone L3.AMA.VDT.P6.03 [1].

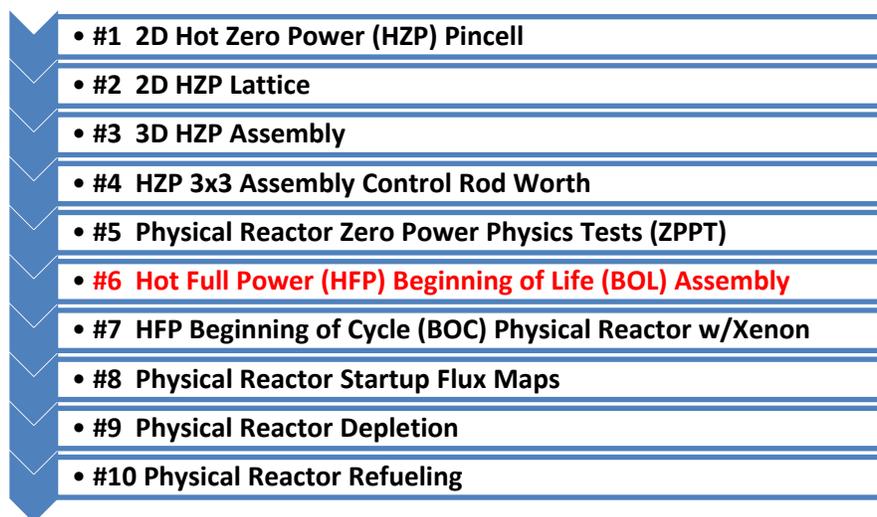


Figure 4.7: The coupled CTF+Insilico capability is being used for AMA progression problem 6.

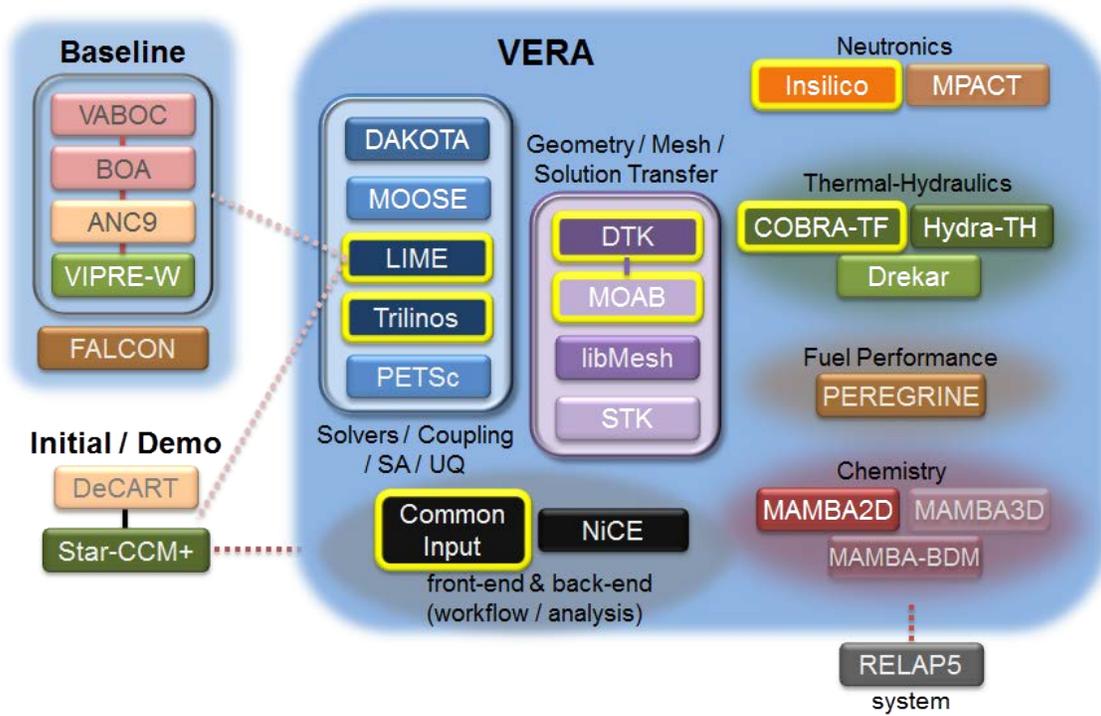


Figure 4.8: Components comprising coupled capability for AMA progression problem 6.

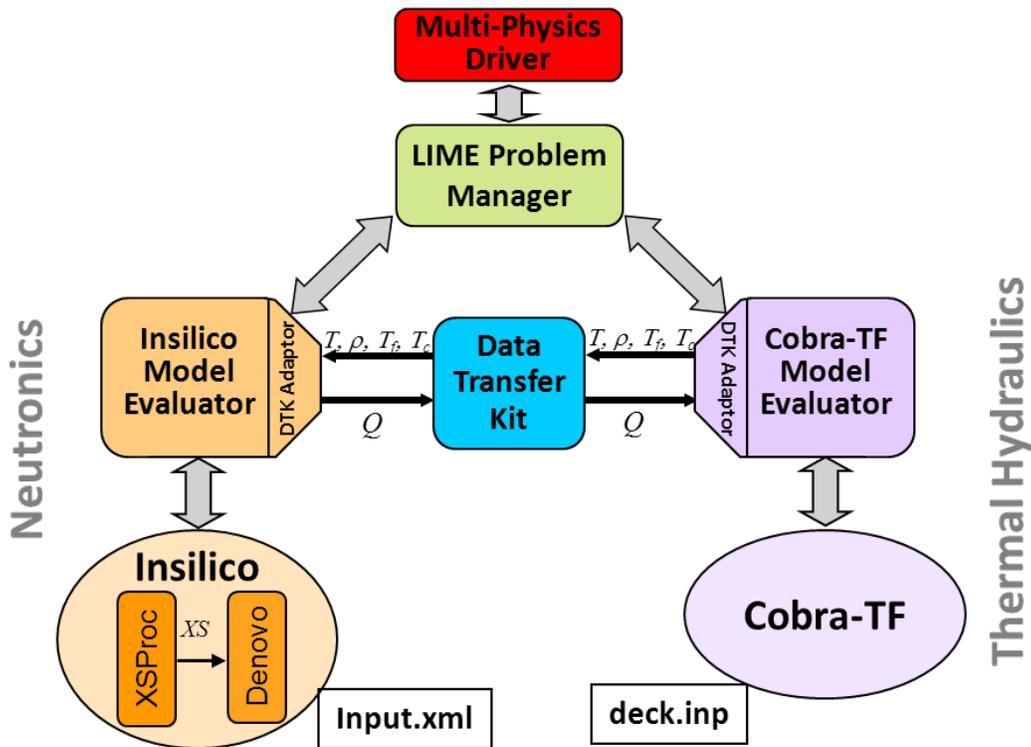


Figure 4.9: Components of the coupled Insilico-CTF application created to solve the example problem.

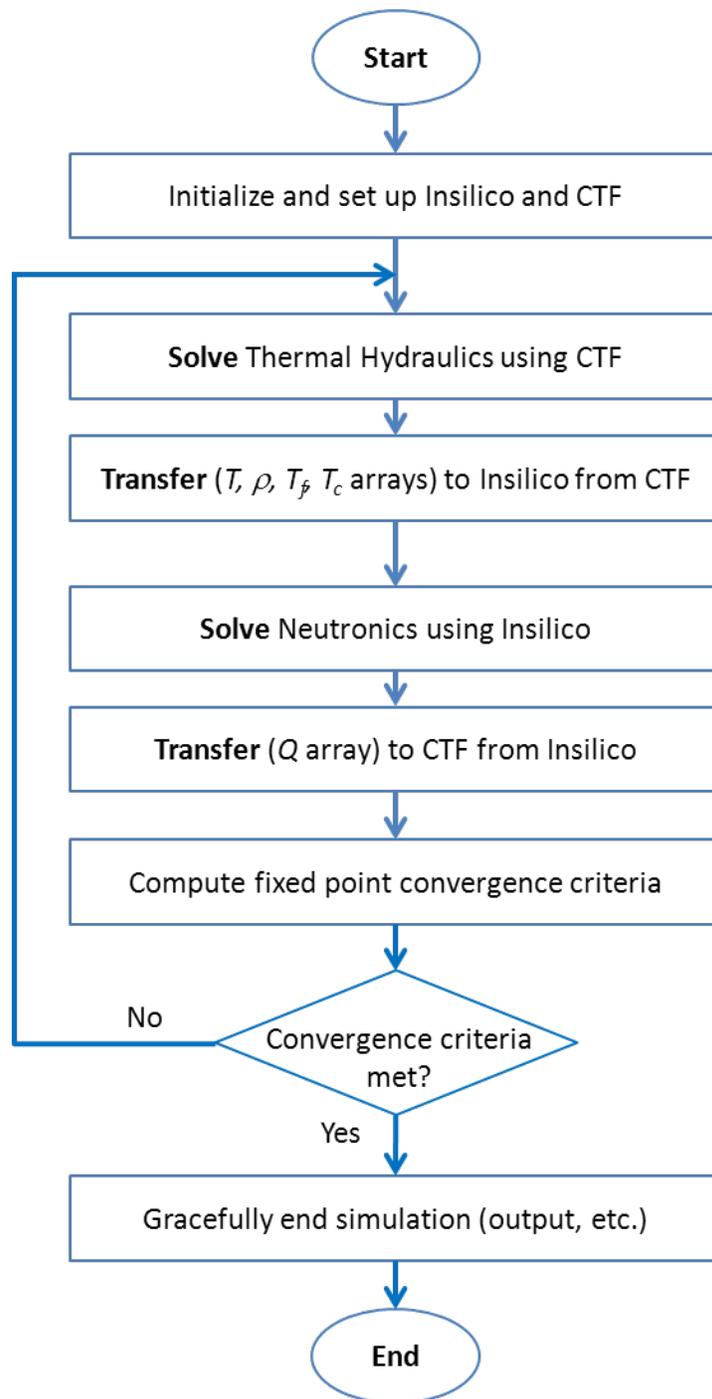


Figure 4.10: Simplified flow chart illustrating the coupled code “Seidel” fixed point algorithm.

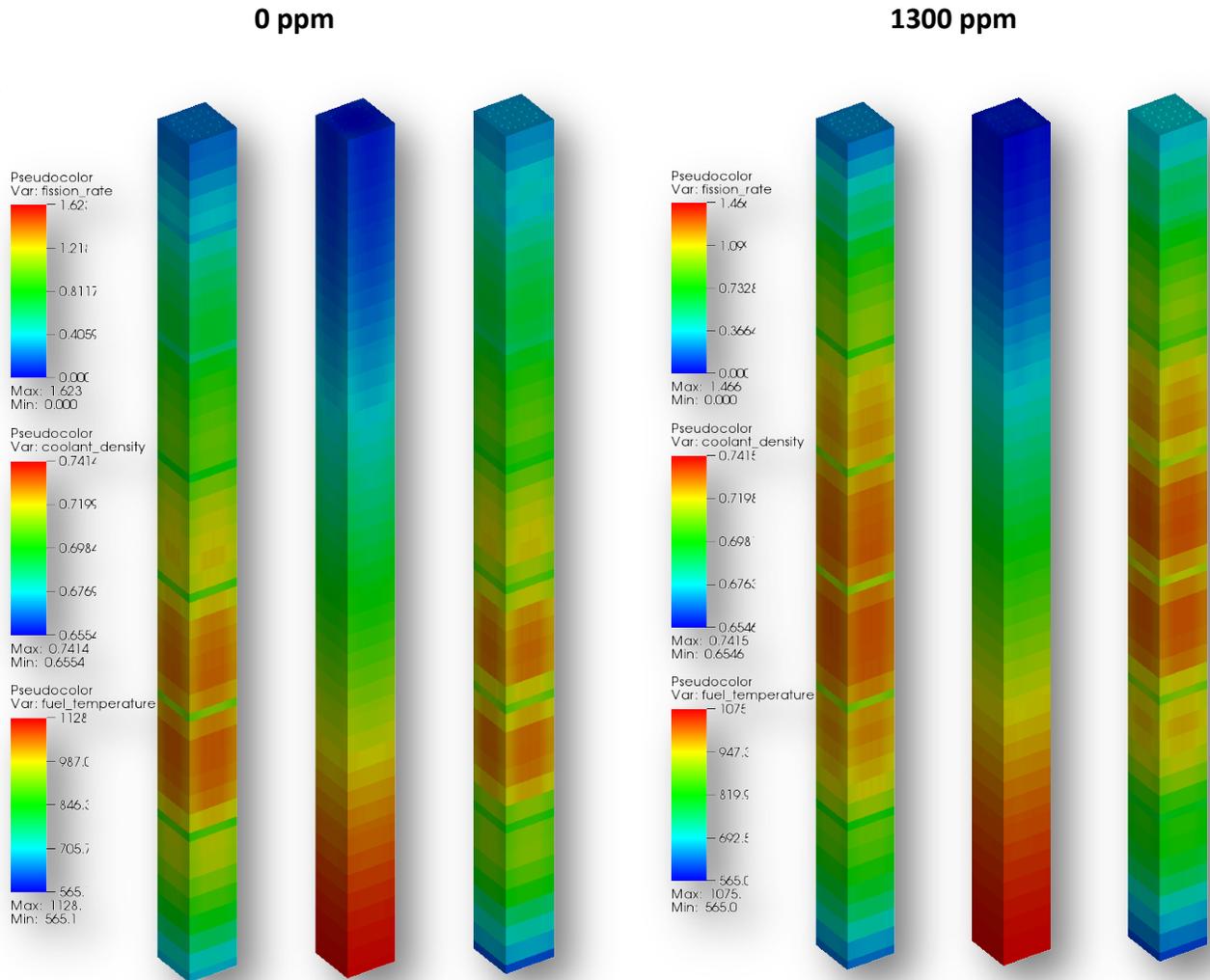


Figure 4.11: Graphical Output of the Fission Rate, Coolant Density, and Fuel Temperatures at 0 and 1300 ppm boron.

Reference milestone report for more detail.

4.6 MPACT

MPACT utilizes the Method of Characteristics (MOC) solution methodology for both 2D and 3D geometries. The Method of Characteristics is a solution to the transport equation in which rays are drawn across the global geometry in discrete angles and the transport equation is integrated along those rays. For extensive detail on the development of MPACT and its capabilities, see the report delivered for RTM L3 milestone RTM.PRT.P6.01 [1].

MPACT has been in the process of being integrated into VERA throughout its development, and progress has been tracked through several PoR-5 milestones (L3:VRI.VERA.P5.08, “Initial MPACT Integration”, L3:VRI.PSS.P5.09, “Improved MPACT Integration”). The current status of the integration of MPACT into VERA is that a repository has been mirrored from the main repository hosted at University of Michigan, LIME drivers have been created to call MPACT subroutines, and MPACT has been placed under continuous and nightly testing in VERA. The next step in the integration is to have MPACT read the VERA common input instead of the standard MPACT input.

The VERA common input processor has two main steps. The first is the standard ASCII input is converted by a PERL script to an XML file. The XML file is then processed by each code to set up the input for the problem. The first step in the process of integrating MPACT is to add a block to the input with MPACT specific data and modify the PERL script to read it. These modifications were made to the VERAIn repository and unit tested.

The next step in the process is to take the new XML file generated by VERAIn and move it into MPACT data structures. The method chosen to do this is to use the Teuchos code in the Trilinos package. Teuchos has the ability to read an XML list and populate its ParameterList data structure. Since Teuchos is written in C++, wrappers were created to provide both C and Fortran access functionality to read and write to the ParameterList. This interface was developed by Roscoe Bartlett of the CASL VRI team with assistance from MPACT developers. The new interface code is named ForTeuchos and is available on the CASL repos in the TeuchosWrappersExt repository.

MPACT uses the ForTeuchos library calls to populate an internal parameter list data structure in MPACT. From this list, the core geometry and solution variables are pulled out of the parameter list. The standard MPACT input file is a text file and is read and processed in two steps, a scan and a read. A similar approach was taken with the XML file and its processor. In the scan step for the XML file, everything from the XML file is read via ForTeuchos subroutines and functions. The necessary data types are converted into Fortran intrinsic data types. The values are then added to the MPACT parameter list, and they are set to global variables where necessary. The material and geometric objects are counted so data objects can be properly allocated. After the file is scanned, there is no need to re-read it since the data is effectively stored. The read step involves processing and initialization of data objects. The cross section library is found and initialized first. Next, materials are processed from the xml format to the MPACT format. This process includes changing material names to cross section library specific isotope identifiers (ZZAAA formatting typically) and converting weight fractions to number densities. Geometry objects are then initialized, converting parameter list information to the necessary geometry object inputs. The order of initialization is pins, modules, lattices, assemblies, and lastly the core. Two current processing assumptions are a default pin mesh is assumed for all pins and ray tracing modules are equivalent to lattices.

The XML and standard MPACT input were able to generate the same results when run for AMA benchmark 2 cases a through d. The geometry for these cases is a 17 by 17 assembly with a 21.5 cm assembly pitch. There are 25 guide tube pins that are filled with moderator, and the rest of the pins are 3.1% enriched UO₂ fuel at a density of 10.257 g/cc. The moderator density is 0.743 g/cc in case A, 0.661 g/cc for cases B through D. The fuel temperatures for cases A through D are 565 K, 600 K, 900 K, and 1200 K respectively. All other material temperatures are 565 K in case A, and 600 K in cases B through D. In the two input types, the pin meshes were the same and the material number densities were equivalent to 5 decimal places. The number densities were

ensured to be similar since MPACT outputs the material data once it is processed for both XML and standard inputs. The assembly gap was modeled consistently, using rectangular pins instead of off-center pin cells. The ray spacing, angular quadrature set, and number of azimuthal and polar angles were consistent between the two input types as well. As seen in Table 4.1, the comparison between the two cases shows very good agreement to convergence criteria.

Table 4.1: Comparison of AMA Benchmark 2 using VERA Common Input.

Case	Eigenvalue		Δk
	XML	Standard	
2A	1.1792851	1.1792857	-0.0000006
2B	1.1801643	1.1801648	-0.0000005
2C	1.1700916	1.1700921	-0.0000005
2D	1.1616291	1.1616295	-0.0000004

The initial integration of the VERA Common Input has been a success. MPACT is able to read a standard lattice and reproduce results that are consistent with the standard MPACT input, and has been now been extended to include the ability to model multiple assemblies and allow for the placement of inserts, control rods, and detectors. 3D capability has been demonstrated for assemblies, and efforts are continuing for larger problems.

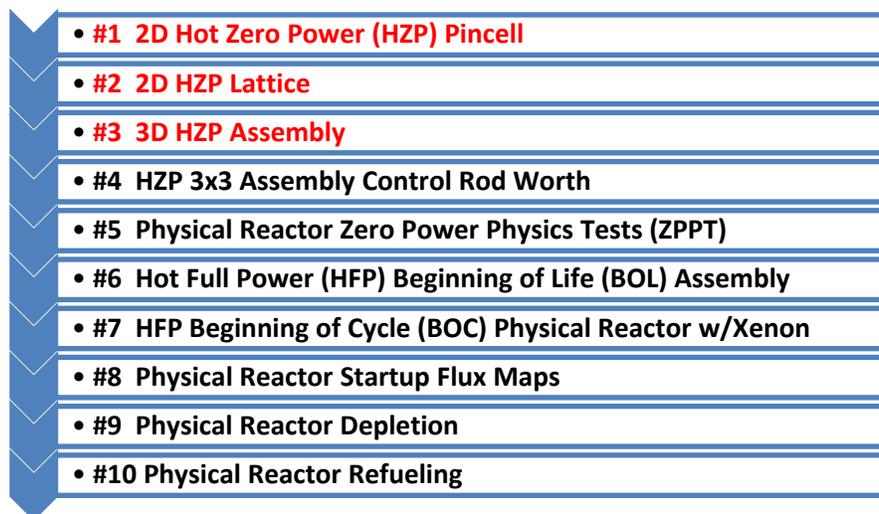


Figure 4.12: Status of MPACT w.r.t. AMA Progression Problems.

References for Section 4.6:

1. B. Collins, B. Kochunas, A. Godfrey, T. Downar, Demonstration of Advanced Pin-Resolved Method of Characteristics (MOC) Capabilities for Neutron Transport, CASL L3 Milestone Report, RTM.PRT.P6.01, 12/21/2012.

4.7 Peregrine2D

A snapshot of the Peregrine source code was transmitted and checked into the CASL repositories prior to the VERA 2.0 snapshot. However, integration effort has focused initially on MOOSE, since it is a required infrastructure component.

References for Section 4.7:

1. D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie´. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. Nucl. Eng. Design, 239, p. 1768–1778, 2009.
2. R. Pawlowski, J. Turner, S. Palmtag, and R. Montgomery, “Initial Demonstration of Peregrine in VERA-CS,” L2.VRI.P7.02, CASL-I-2013-0165-000, July 31, 2013.

4.8 Tiamat: COBRA-TF + Insilico + Peregrine2D

Tiamat provides an initial integration of the 2D axisymmetric fuel rod modeling functionality of Peregrine with VERA-CS components to provide an improved capability for AMA progression problem 6. Figure 4.14 depicts the VERA components comprising Tiamat, Figure 4.15 illustrates the data transfers performed for a coupled simulation, and Figure 4.16 shows the MPI communication layers implemented in Tiamat. A significant aspect of Tiamat is its use of separate blocks of processors for each physics component, allowing more flexible allocation of computational resources.

Figure 4.17 and Figure 4.18 show initial results using Tiamat. For more implementation details and further results, see the milestone report for L2.VRI.P7.02 [1].

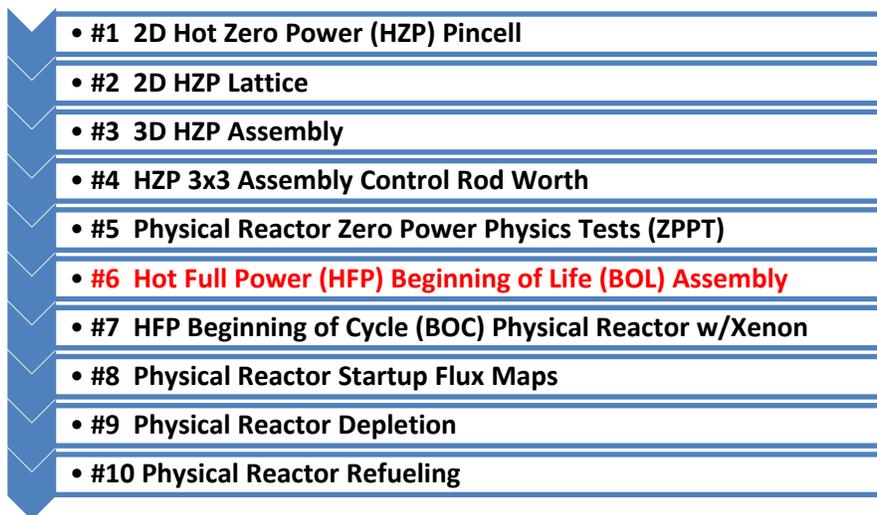


Figure 4.13: Tiamat provides additional capability for AMA Progression Problem 6 and beyond.

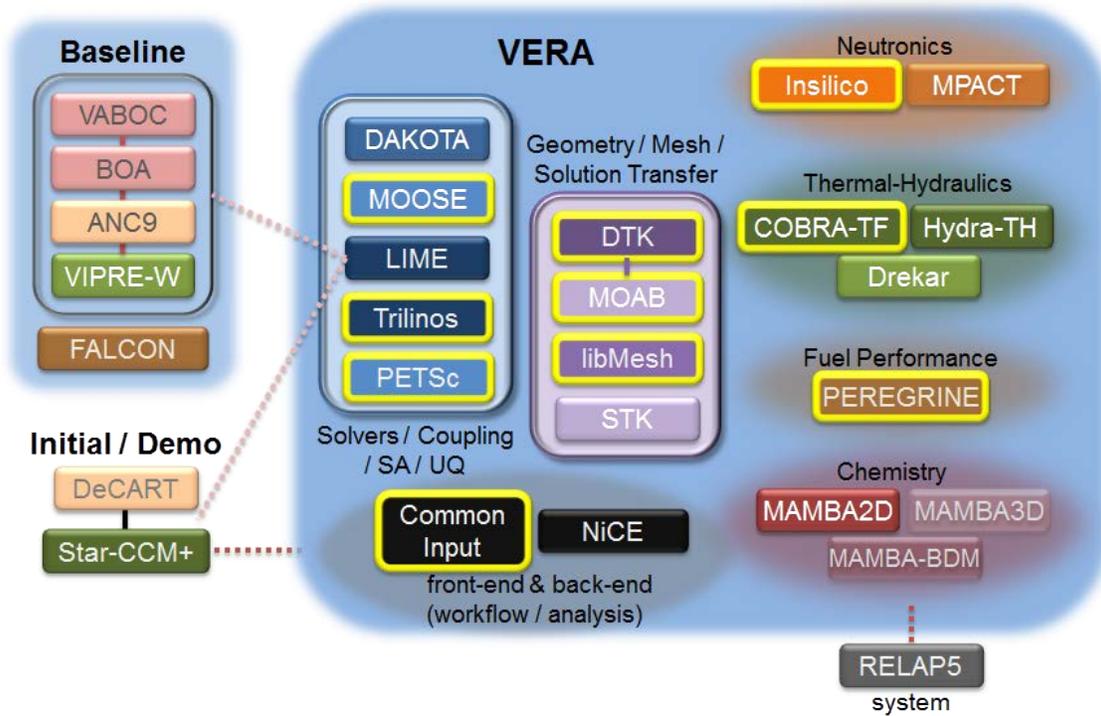


Figure 4.14: VERA Components comprising Tiamat.

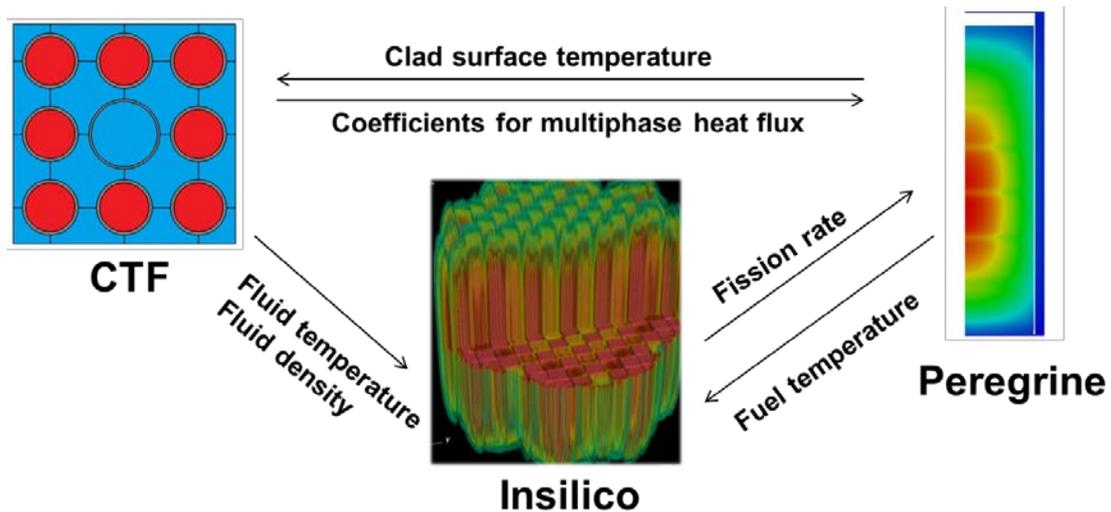


Figure 4.15: Data transfers between physics components in Tiamat.

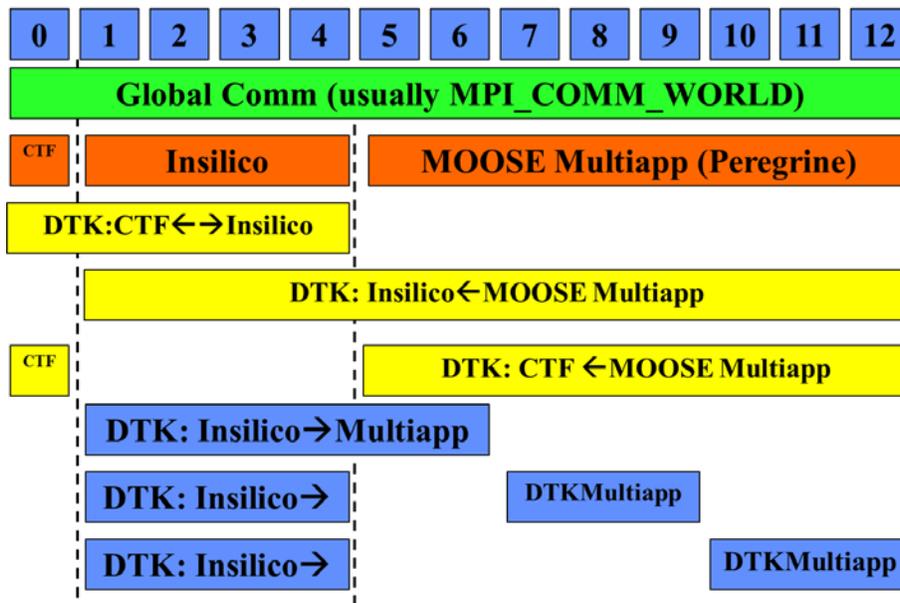


Figure 4.16: Depiction of the MPI communication layers in Tiamat.

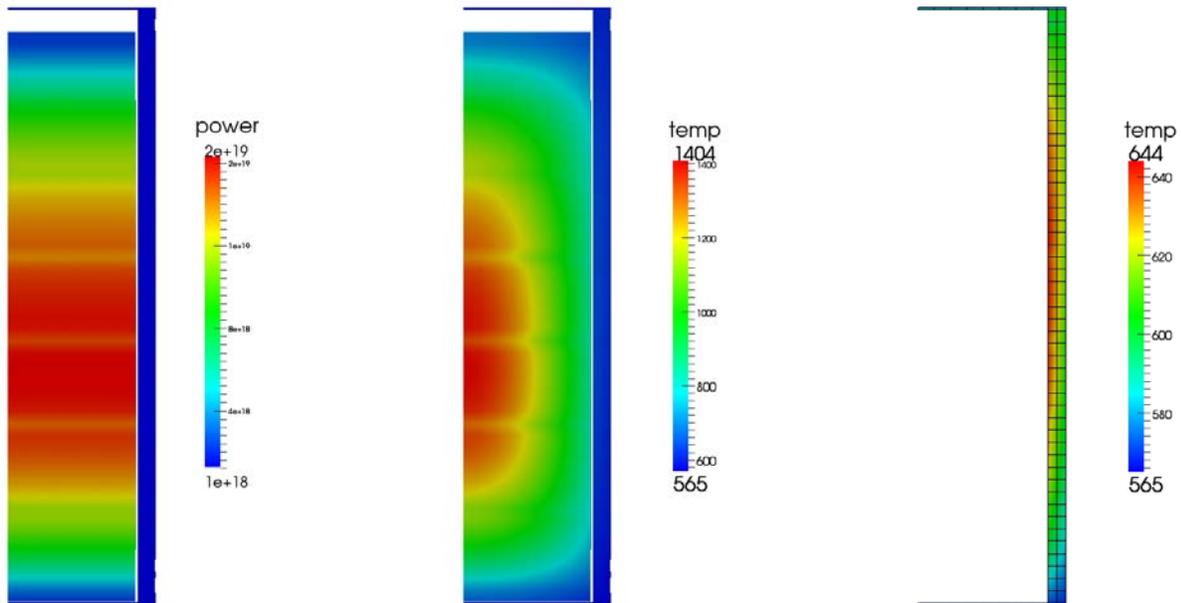


Figure 4.17: Surface plots of fission rate (from Insilico) and temperature in Peregrine. The plot on the right is scaled to show clad temperatures.

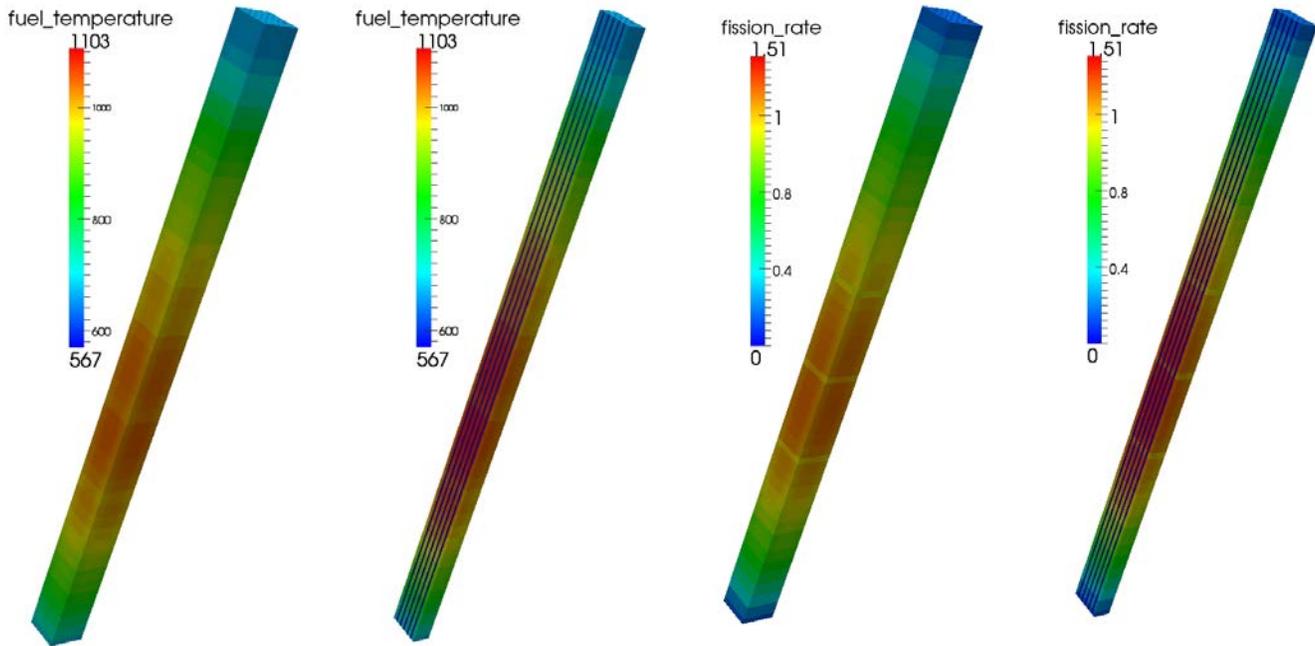


Figure 4.18: In-silico averaged fuel temperature and fission rate.

References for Section 4.8:

1. R. Pawlowski, J. Turner, S. Palmtag, and R. Montgomery, "Initial Demonstration of Peregrine in VERA-CS," L2.VRI.P7.02, CASL-I-2013-0165-000, July 31, 2013.

5 OTHER VERA COMPONENTS

5.1 Hydra-TH

Hydra-TH refers to the hybrid finite-element/finite-volume incompressible/low-Mach Navier-Stokes flow solver in the Hydra toolkit that is being used for CASL thermal-hydraulics applications [1]. Hydra-TH is built as one of a number of virtual physics using the Hydra multiphysics toolkit. The Hydra toolkit is written in C++ and provides a rich suite of components that permits rapid application development, supports multiple discretization techniques, provides I/O interfaces to permit reading/writing multiple file formats for meshes, plot data, time-history, surface-based and restart output.

The Hydra-TH theory manual [2] presents a detailed theoretical background. All transport variables are cell-centered and treated with a conservative discretization that includes a high-resolution monotonicity-preserving advection algorithm. The spatial discretization is formally derived using a discontinuous-Galerkin framework that, in the limit, reduces to a locally-conservative finite-volume method. The high-resolution advection algorithm is designed to permit both implicit and explicit advection with the explicit advection targeted primarily at volume-tracking with interface reconstruction. The time-integration methods include backward-Euler and the neutrally-

dissipative trapezoidal method. The coding for an optional BDF2 time-integrator has also been provided, but is not currently used in Hydra-TH. The implicit advective treatment delivers unconditional stability for the scalar transport equations, and conditional stability for the momentum transport equations. A sharp stability estimate for the momentum equations is not tractable, but operational experience shows that the algorithm is stable for $20 \leq CFL \leq 40$. For steady-state problems, backward-Euler provides additional damping that, in conjunction with $20 \leq CFL \leq 40$, provides a computationally efficient solution method. For URANS and LES computations, the trapezoid rule is neutrally dissipative, and delivers optimal performance for the more moderate CFL requirements for transient flow.

The solution algorithm used in Hydra-TH is based on a second-order incremental projection algorithm. Projection methods are the most computationally efficient solution method available for solving the time-dependent Navier-Stokes equations.

In order to address fluid-structure problems, Hydra-TH uses an arbitrary Lagrangian-Eulerian (ALE) formulation and provides a mesh-deformation interface that can support multiple different mesh smoothing algorithms. Details on the ALE formulation may be found in the Hydra-TH theory manual. The added-mass terms are computed for the structural coupling and can be exported for any structural solver. For explicit coupling, Hydra-TH provides a pressure-stabilized algorithm based on Niche's variational method that circumvents the stability limitations associated with highly flexible structures and near unity fluid/solid density ratios. For conjugate heat transfer, there are multiple alternatives available in Hydra-TH that include explicit coupling with third-party heat conduction solver, internal coupling using the existing heat conduction solver supported by the Hydra toolkit and the multiphysics manager, or direct integration (with continuous meshing). The calculation of exported fields for both fluid-structure interaction and conjugate heat transfer are implemented for explicit coupling methods, and can be easily extended for use in tightly-coupled solution strategy. It is anticipated that driving CASL applications will determine the most suitable FSI/CHT solution strategy.

References for Section 5.3:

1. M. A. Christon, R. Lowrie, N. Barnett, J. Bakosi and M. Francois, Hydra-TH L3 Milestone (THM.CFD.P3.02), CASL L3 Milestone Report, September 30, 2011.
2. M. A. Christon, Hydra-TH Theory Manual, Tech. Rep. LA-UR 11- 05387, Los Alamos National Laboratory, September 2011.

5.2 MAMBA2D

MAMBA2D v2.0 is a 3D Fortran computer code developed to model CRUD formation and growth for CASL. In MAMBA a general time-dependent 3D heat transport equation is solved to obtain the temperature distribution throughout the crud layer on a single pin (or on selected regions of a single pin). The heat transport solution also includes the localized heat sinks due to regions of sub-cooled nucleate boiling (SNB) which may be occurring inside the crud deposit. The rod heat flux at the cladding surface and coolant temperature (or flux) at the crud surface represent the external boundary conditions. These can be supplied by the user using MAMBA supplied default models or through external coupling to DeCART and Star-CCM+ (via file transfers) or advanced thermal hydraulics and neutronics codes in the future.

The local crud thermal conductivity varies in both space and time due to the changes in porosity. The change in porosity is due to the internal deposition of nickel ferrite (NiFe_2O_4) within the pores of the crud which slowly fills them. The local porosity can also change more quickly due to the precipitation of lithium-tetraborate ($\text{Li}_2\text{B}_4\text{O}_7$). This is the primary mechanism for boron deposition inside the crud.

The deposition of nickel ferrite and lithium-tetraborate within the crud are both enhanced by localized SNB which generates significant Darcy flow within the porous crud layer. Both the Darcy flow and diffusion within the crud layer are modeled and these mass transport mechanisms lead to significant increases in the local concentrations of the various soluble species (Ni, Fe, Li and boric acid) inside the crud.

Advanced chemistry/thermodynamic models have also been incorporated into MAMBA v2.0 for treating the coolant chemistry (Li, B, H_2 , Ni, Fe, $\text{B}(\text{OH})_3$, and several ionic species) and for accurately determining the precipitation parameters. These models are based on the>NNL/BOA developed mechanisms and equilibrium constant correlations. The advanced chemistry/thermodynamic models are applied at both the crud/coolant interface as well as each internal volume element inside the crud at each time step. The precipitation parameter determines when a given volume element begins to rapidly fill with lithium-tetraborate.

Currently, the recommended values from the Westinghouse WALT Loop study [1, 2] are used for the various crud properties: porosity, thermal conductivity, chimney density, and chimney radius. An adaptive 3D mesh is used which “grows” the 2D crud surface as mass deposits onto the surface elements. The surface particulate deposition rate is governed by two rate parameters: one for non-boiling regions and one for boiling regions. The boiling deposition rate is multiplied by the local mass evaporation (steaming) flux leaving the crud surface via the chimneys. Thus, local SNB within the crud layer leads to enhanced particulate deposition and crud growth at the surface above.

References for Section 5.2:

1. Wang, G.; Byers, W.A.; Karoutas, Z.E.; Young, M.Y.; Jacko, R.J. [WEC]; Hochreiter, L.E. [Penn. State Univ.], “Single Rod Heat Transfer Tests to Study the Effects of Crud Deposition,” 14th Intl. conf. on nuclear engineering (ICONE 14), 17-20 Jul 2006.
2. G. Wang, W. A. Byers, M. Y. Young, and Z. E. Karoutas, “Westinghouse Advanced Loop Tester (WALT) Update,” ASME Conf. Proc. 2008, 469 (2008), DOI:10.1115/ICONE16-48480.

5.3 COBRA-TF + MAMBA2D

The 2D CRUD deposition capability described in Section 5.2, MAMBA2D, is being coupled to the subchannel thermal-hydraulic component CTF (Section 4.3) for analysis of CRUD-related challenge problems such as CRUD-induced power shift (CIPS). VERA components involved in this capability are shown in Figure 5.1. Although this capability remains in development, coupled results have been demonstrated up to a 3-rod by 3-rod sub-assembly. This calculation consists of 36 independent MAMBA2D calculations in conjunction with the standard CTF calculation.

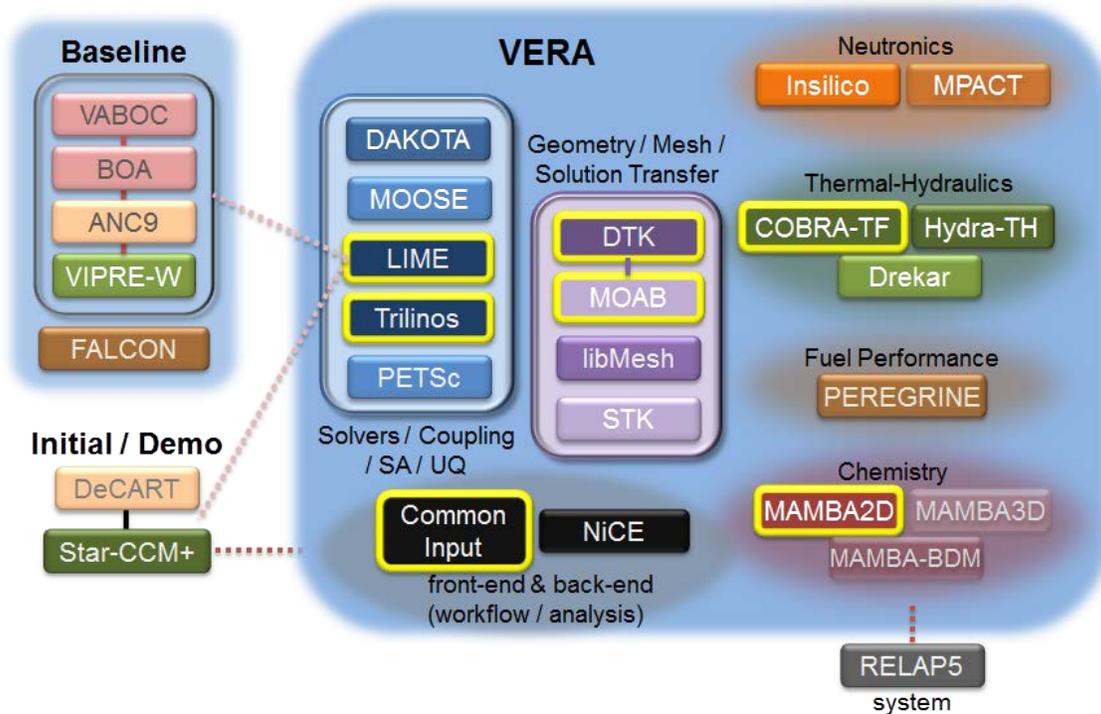


Figure 5.1: VERA components comprising coupled CTF+MAMBA2D capability.

5.4 RELAP5-3D

RELAP5-3D is used for best-estimate transient simulation of light water reactor coolant systems during postulated accidents. It models the coupled behavior of the reactor coolant system and the core for loss-of-coolant accidents and operational transients such as loss of offsite power, loss of feedwater, and loss of flow. Control system and secondary system components are included to permit modeling of plant controls, turbines, condensers, and secondary feedwater systems. The anticipated use for RELAP5-3D in VERA is to model the reactor primary system outside the pressure vessel and the secondary systems.

In VERA 1.0, the RELAP5-3D model evaluator (needed for wrapping within LIME) was designed to run a stand-alone simulation straight through from start time to end time as specified in the RELAP5-3D input file. This mode of operation did not allow LIME to control when a time step was taken or what the time step size would be; the RELAP5 input model controlled those parameters. Because of these limitations, RELAP5-3D was not yet able to operate in a coupled fashion with other codes.

The RELAP5-3D internal time step loop iterates on the basis of attempted advancements. For each attempted advancement, several checks on solution acceptability are performed, testing for conditions such as excessive mass error, material or thermodynamic fluid properties out of range, phase appearance/disappearance, large pressure changes in a volume when non-condensable appear, etc. When the attempted advancement is determined to be unacceptable, the advancement is repeated with the same time step size or with reduced time step size, depending on the nature of the failure.

In VERA 2.0, the RELAP5-3D model evaluator has been modified to couple more tightly to the LIME problem manager. It contains changes to the RELAP5-3D advancement logic to properly account for time step size reductions, increases, and time step repeats and return control to LIME only when the advancement has been successful. Additionally, RELAP5-3D can accept the time step size requested by LIME instead of using the internal time step size control logic. These developments are the key changes needed so that in the future it could be used in a coupled capability. The CASL version of RELAP5-3D is currently fully integrated as a standalone application within VERA and has the same capabilities as version 3.0.0 of the code developed at INL and available to members of the International RELAP5 Users Group.

5.5 Drekar

Drekar is a large-scale parallel, fully-implicit stabilized finite element code for low-Mach turbulent CFD developed at Sandia National Laboratories [1]. The code has the capability to include fully coupled heterogeneous multiphysics formulations such as fluid flow, conjugate heat transfer and chemistry. It provides an advanced fully-implicit T-H capability for VERA and has been used for high resolution T-H simulations of the 3x3 rod geometry for the CASL GTRF challenge problem.

Recent Drekar development activities include the following:

- Drekar is completely integrated into the VERA build, continuous integration, and nightly test system. Drekar can be run stand-alone through the LIME/VRIPSS drivers.
- Input for the Drekar code is handled through a Teuchos parameter list [2]. This parameter list is typically generated by a specialized XML parser. The previous XML/parameter list implementation required some overly verbose and possibly error prone structures to exist in the Drekar input file. A number of tasks were completed to improve the capabilities of the Teuchos parameter list and significantly simplify the Drekar input XML file. An order preserving parameter list was implemented. Better error reporting for mismatched XML entries was added. Error checking for accidentally duplicating a sublist in XML was added. The XML input file is now more robust and clear.
- A multiphysics conjugate heat transfer capability was implemented and demonstrated in Drekar under the L3 milestone VRI.PSS.P4.02 [3]. As part of this milestone, a number of issues were found and documented in Appendix 3 of the milestone report related to the data transfer toolkit.
- In terms of physics, the most recent release of Drekar includes models for laminar constant density incompressible flow (Navier-Stokes), turbulent constant density LES (WALE and VREMIN models), and turbulent URANS (SARANS, k-epsilon is optional). This release does not include any adjoint capabilities, or the uncertainty quantification capabilities. Initial verification and validation of the various models was undertaken as part of the Drekar release under milestone L2 THM.P4.02 [4] and is documented in the theory manual [5].

References for Section 5.2:

1. T. M. Smith, J. N. Shadid, R. P. Pawlowski, E. C. Cyr, and P. D. Weber, Reactor Core Sub-Assembly Simulations Using a Stabilized Finite Element Method, in The 14th International Topical Meeting on Nuclear Reactor Thermalhydraulics, NURETH-14, 2011.
2. <http://trilinos.sandia.gov/packages/teuchos/>

3. R. Pawlowski, Level 3 Milestone Deliverable - VRI.PSS.P4.02, Feb. 9, 2012
4. J. Shadid, R. Pawlowski and T. Smith, L2 THM.P4.02 milestone report, (in preparation)
5. R. Pawlowski, Drekar Theory Manual. (in preparation)

5.6 Insilico-Drekar

Coupling of the Denovo-based Insilico neutronics capability and the Drekar CFD capability was completed and demonstrated in PoR-6. Although Drekar has been de-emphasized within CASL in favor of Hydra-TH, this capability provides a demonstration of advanced coupling that can be leveraged for a future Insilico-Hydra-TH capability.

Details on the Insilico-Drekar capability can be found in documentation of the PoR-5 L3 milestone VRI.PSS.P5.07, “DataTransferKit-based neutronics-TH coupling” [1].

References for Section 5.3:

1. https://casl-dev.ornl.gov/trac/casl_milestones/ticket/649

6 VERA PHYSICS CAPABILITIES UNDER DEVELOPMENT

Components that have been checked into the CASL VRI repository but have not yet been incorporated into the official VERA build and test environment. Each of these codes is expected to play important roles in future releases of VERA and are expected to be fully incorporated in the near future.

6.1 COBRA-TF + MPACT

CTF is being coupled to MPACT (Section 4.6) in order to provide a pin-resolved neutronics capability for AMA progression problem 6 (and beyond). VERA components comprising this capability are depicted in Figure 6.1, and are similar to those used for the CTF+Insilico capability described in Section 4.5.

The pin-wise T/H feedback capability has been initially implemented for simple problems and is currently undergoing testing. The pin-wise feedback in MPACT only maps the active fuel regions including guide tubes. Below the active fuel the inlet conditions are applied uniformly and above the active fuel the exit (or plenum) conditions are applied uniformly. For radial reflectors outside the fuel, a core average quantity for temperature and coolant density is applied uniformly. Interfaces that MPACT was supplying to DTK have been refactored and verified for the above stated capability. Instead of having ~20 interfaces there are now 5, and now include the return of error codes to the client so that DTK can throw exceptions for errors within MPACT. An interface for a destructor was also added. Once the testing of the data transfers is complete for a pseudo-TH feedback system, testing with coupling to CTF will be performed.

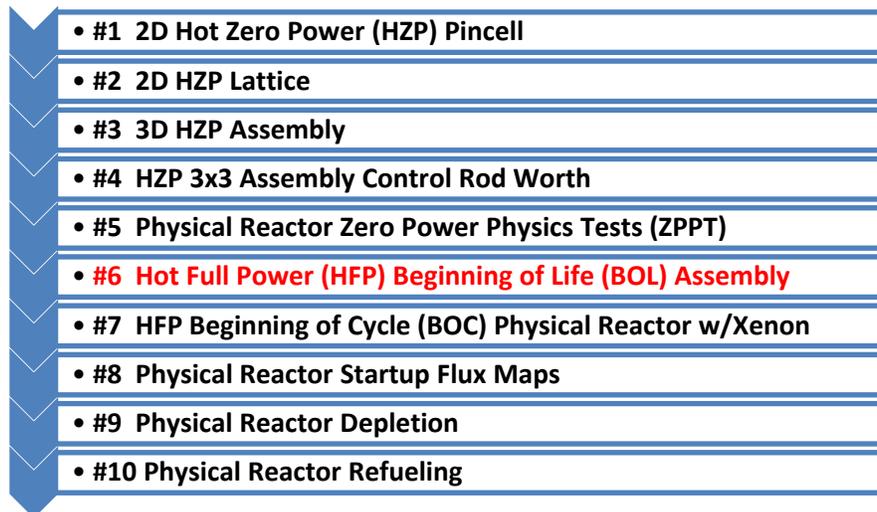


Figure 6.1: CTF+MPACT provides a pin-resolved capability for AMA progression problem 6.

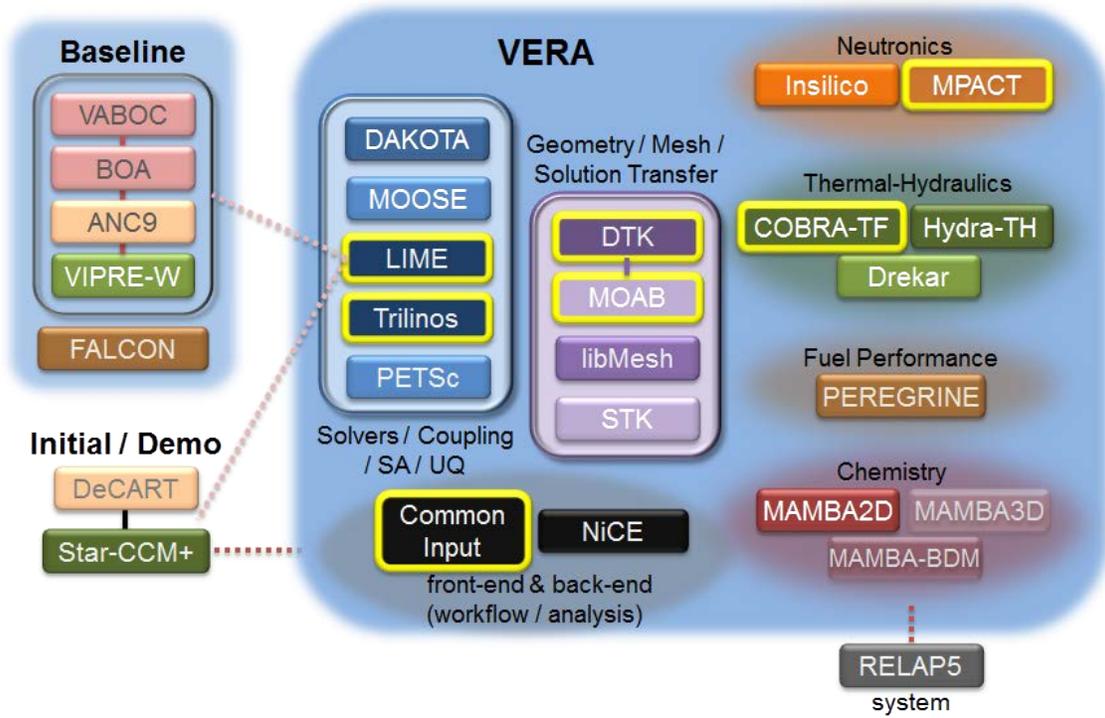


Figure 6.2: VERA components used in an initial pin-resolved capability for AMA progression problem 6.

7 VERA BASELINE COMPONENTS

7.1 ANC9.5

ANC9 [1] is a Westinghouse developed multi-dimensional nodal diffusion code for nuclear core design calculations. It predicts core reactivity, assembly power, rod power, detector thimble flux, and other relevant core characteristics. Version 9.5 contains several important updates and capabilities required for use in the improved ANC-VIPRE-BOA coupling strategy for CASL.

7.2 VIPRE-W

VIPRE-W [2] is a Westinghouse version of the VIPRE-01 code. VIPRE-01 is a thermal-hydraulic sub-channel code based on the COBRA codes developed by Pacific Northwest National Laboratories under sponsorship of the Electric Power Research Institute (EPRI). VIPRE-W contains enhancements for PWR applications, including the mass evaporation and grid spacer heat transfer models required for CIPS risk assessment.

7.3 DAKOTA-VIPRE-W

The integration of DAKOTA described in Section 3.5 has been used to perform sensitivity studies on the Westinghouse subchannel analysis capability, VIPRE-W, as shown in Figure 7.1. Results of these studies appear in documentation of CASL L3 Milestone VUQ.P2.03.

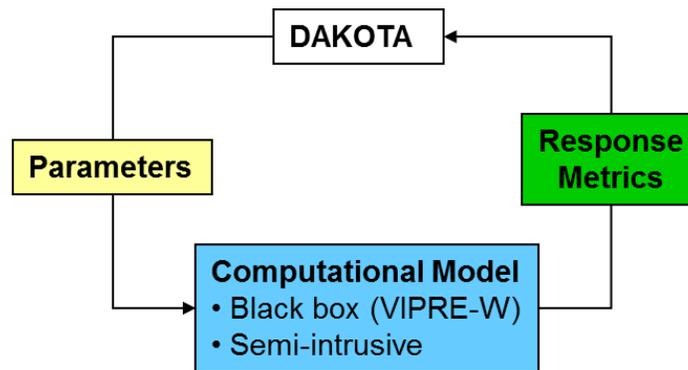


Figure 7.1: Data transfer for DAKOTA-VIPRE-W.

Subsequent to these initial studies, DAKOTA is being integrated more fully into VERA in order to perform sensitivity analysis and uncertainty quantification studies.

7.4 ANC9.5-VIPREW (ANCLVIPRE)

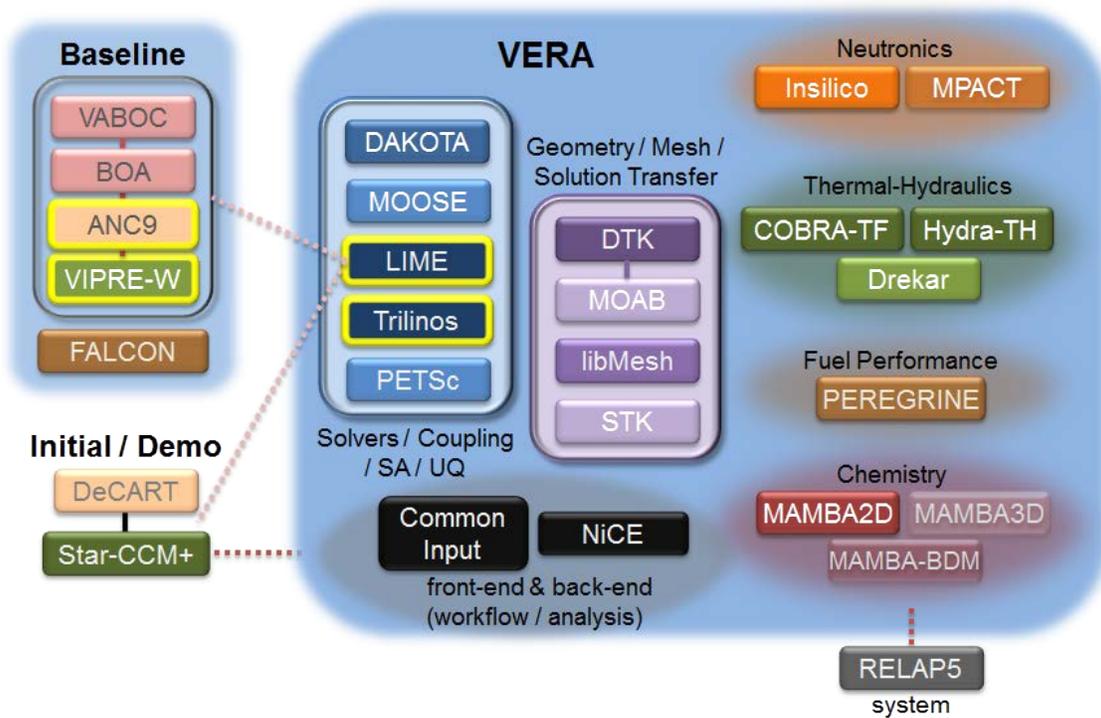


Figure 7.2: Components used in ANC-VIPRE-W coupled baseline capability (ANCLVIPRE).

Prior to CASL, Westinghouse had developed a loose coupling of ANC and VIPRE-W known as ANCKVIPRE. This capability was loosely-coupled to RETRAN to create RAVE.

As a first step in development of the coupled capability described in Section 7.5, ANCKVIPRE was re-factored as a LIME-based coupled capability known as ANCLVIPRE. Further details can be found in Section 7.5.

7.5 ANC9.5-VIPREW-BOA

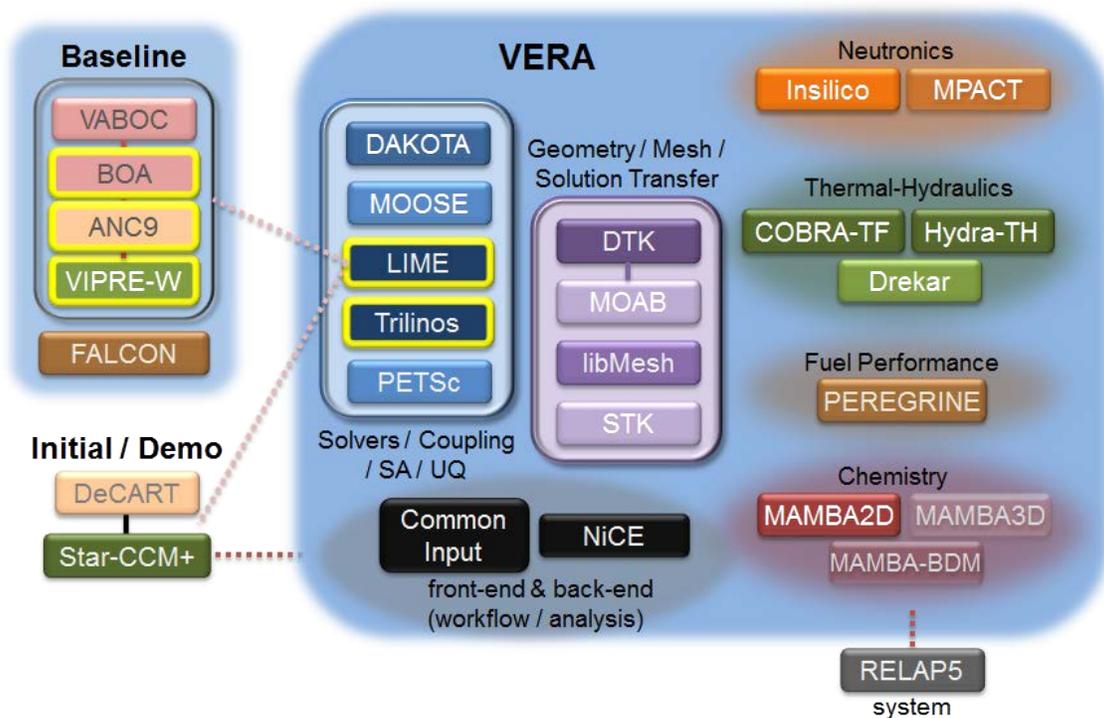


Figure 7.3: Components used in ANC-VIPRE-W-BOA coupled baseline capability.

BOA [3] is the Boron-induced Offset Anomaly Risk Assessment Tool. This is a proprietary code developed and maintained by EPRI. The tool is used to assess the risk of axial offset anomaly (AOA) by simulating boron-deposition in the sub-cooled nucleate boiling region of high-duty fuel.

A revised ANC-VIPREW-BOA multiphysics simulation capability has been developed and tested for CASL. This uses a time-consistent coupling strategy (based on a 2nd order Crank-Nicolson time integration scheme) and significantly improves this CASL baseline simulation capability for time-dependent depletion calculations. As noted above, developing this capability required the adoption and use of a newer version (9.5) of ANC. Details of the time-consistent coupling approach are described in [4], but some key aspects are provided here.

Figure 7.4 shows a LIME-based coupling diagram for ANC9.5-VIPREW-BOA. The coupling strategy originally employed (in VERA 1.1) did not involve a full tight coupling of all three codes, but had an explicit time-lagged coupling to the BOA code.

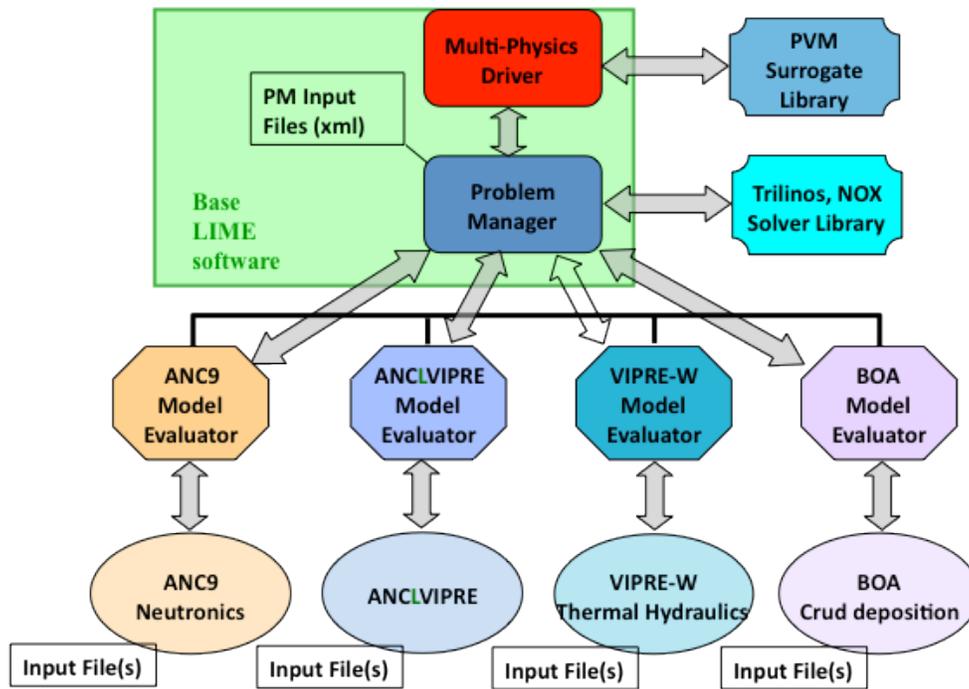


Figure 7.4: LIME-based Code-Coupling Diagram for ANC9.5-VIPREW-BOA.

7.5.1 Multiphysics Coupling Diagram

Figure 7.5 illustrates the coupling dependencies between the different physical processes modeled in ANC9.5-VIPREW-BOA from a data-flow perspective. The different colored objects represent the four major physical processes being modeled by the physics codes. In this diagram it is important to distinguish between the time dependent physics (represented by ovals), and the steady state physics (represented by rectangles). Note that to compute the instantaneous depletion rate a depletion model/code (colored blue) needs the neutron flux values to be externally provided. Also note that to compute the boron precipitation rate the boron code (colored green) needs the current thermal hydraulic state to be externally provided. However, neither of these rate-of-change quantities is used until advancing the system to a new time state.

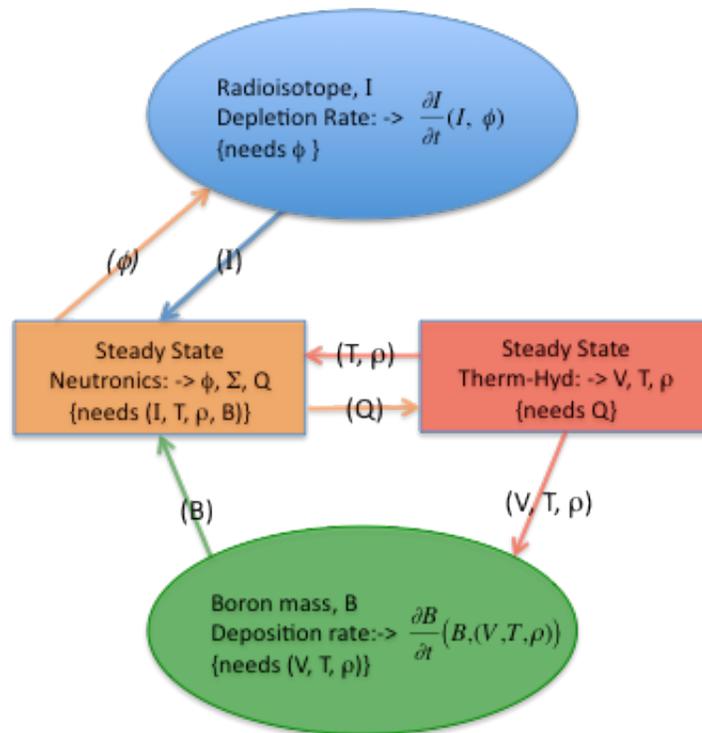


Figure 7.5: A physics-focused coupling diagram showing data dependencies and data exchange paths.

For the depletion problems being modeled by ANC9.5-VIPREW-BOA, a reactor can be usefully modeled as if it exists in a series of pseudo-stationary states during which the neutron production and loss terms are balanced. In the context of the physics codes, each of these reactor states are characterized by the current values and distribution of “state variables” such as the radioisotopes “I”, the boron mass “B”, the neutron flux f , the neutron cross sections S , local power Q , temperatures T , coolant densities ρ , and coolant velocities V (or, equivalently, mass flow rates).

To advance in time ANC9.5-VIPREW-BOA now uses a method which preserves the coupling dependencies noted above in a time-consistent fashion. By time-consistent we mean that the information passed between codes (including the coupling dependencies) are all associated with the same time-state (e.g., one set is not time lagged from another).

7.5.2 Time advancement of the coupled multiphysics system

Figure 7.6 illustrates data flow between the different physics models for two time steps, “n” and “n+1”, and will be referenced here to help explain the time-advancement approach implemented.

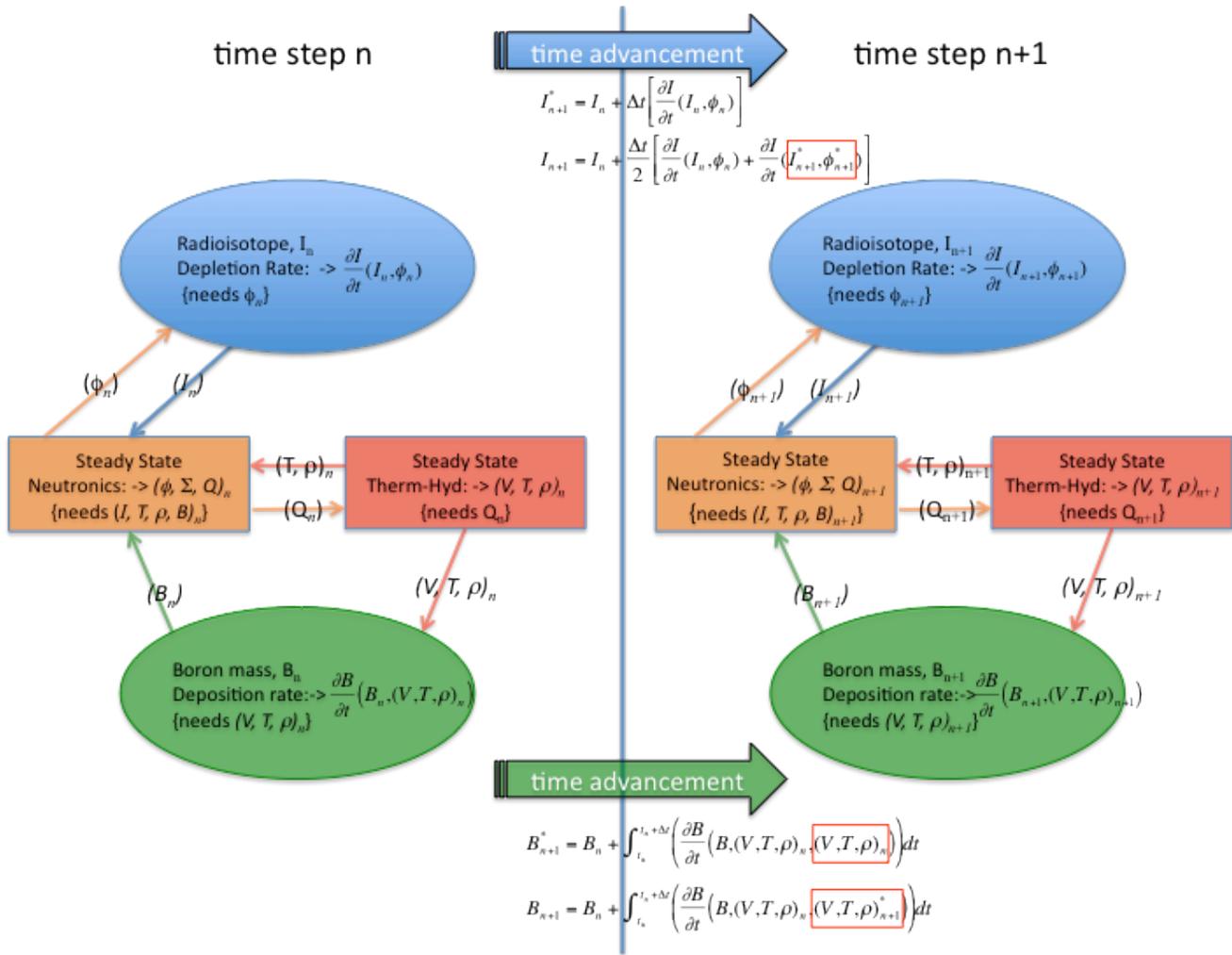


Figure 7.6: Data-flow diagram for a time-consistent multi-physics time advancement scheme based on a predictor-corrector method.

The goal of each integration step is to advance the system state from time step “n” to “n+1”, where all quantities of interest are initially known at time step n. This involves the time-advancement of two time-dependent physics: (1) the depletion of radioisotopes (treated within the ANC9.5 code), and (2) the deposition of boron in the form of CRUD (modeled by the BOA code). A two-step predictor-corrector approach to the time advancement of these equations is first described.

An explicit predictor step for the depletion physics modeled in ANC¹ will be represented here as

¹ Since isotope depletion is chain behavior, ANC must effectively model transmutation equations of form

$$\frac{dN_i}{dt} = -(\sigma_i \phi + \lambda_i) N_i + Y_i$$

$$I_{n+1}^* = I_n + \Delta t \left[\frac{\partial}{\partial t} (I_n, \phi_n) \right] \quad (3)$$

where I denotes a vector of radioisotopes, $\frac{\partial}{\partial t} (I_n, \phi_n)$ is the rate of change of I with respect to time, and ϕ denotes the neutron fluxes. This is a significant simplification of some more complex equations, but is adequate for the purposes of describing the coupling issues in the time-advancement problem.

A predictor step for the crud deposition modeled in BOA can be represented as

$$B_{n+1}^* = B_n + \int_{t_n}^{t_n+\Delta t} \left(\frac{\partial B}{\partial t} (B, (V, T, \rho)_n, (V, T, \rho)_n) \right) dt \quad (4)$$

where B denotes the boron mass distribution and “ (V, T, ρ) ” denotes the thermal hydraulic state of the system computed in VIPRE-W. Two things must be noted here. First, because BOA integrates the boron deposition models in time using a set of smaller time steps, we represent that here as an integral from time t to $t+\Delta t$. Second, BOA requires, as input, the thermal hydraulic state of the system both at time t and at $t+\Delta t$. For the predictor step we keep the thermal hydraulic state constant, i.e., we input the same $(V, T, \rho)_n$ values both at time t and at time $t+\Delta t$.

Before the corrector step for the depletion physics can be solved, values for ϕ_{n+1}^* (the neutron fluxes based on I_{n+1}^*) must be computed. As indicated by Figure 2, this is done by solving the steady-state coupled (Thermal Hydraulics) \leftrightarrow (Neutronics) problem based not only on I_{n+1}^* but also on B_{n+1}^* . The solution of this coupled problem yields values for ϕ_{n+1}^* , to be used in the depletion physics in the corrector step:

$$I_{n+1} = I_n + \frac{\Delta t}{2} \left[\frac{\partial}{\partial t} (I_n, \phi_n) + \frac{\partial}{\partial t} (I_{n+1}^*, \phi_{n+1}^*) \right] \quad (5)$$

It also yields values for $(V, T, \rho)_{n+1}^*$, which are needed by the boron deposition physics in its corrector step:

$$B_{n+1} = B_n + \int_{t_n}^{t_n+\Delta t} \left(\frac{\partial B}{\partial t} (B, (V, T, \rho)_n, (V, T, \rho)_{n+1}^*) \right) dt \quad (6)$$

When both the depletion and boron predictor equations are solved, one final step is required. This final step is to solve the steady-state coupled (Thermal Hydraulics) \leftrightarrow (Neutronics) problem based on I_{n+1} and B_{n+1} so that all quantities of interest are now known at time step $n+1$.

for a whole chain instead of computing “rate-of-change” for each individual isotope. In the modeling approach used by ANC, important parts of the problem are “precalculated” by another code (PHEONIX) and results tabulated for use by ANC. Additional details are also omitted here.

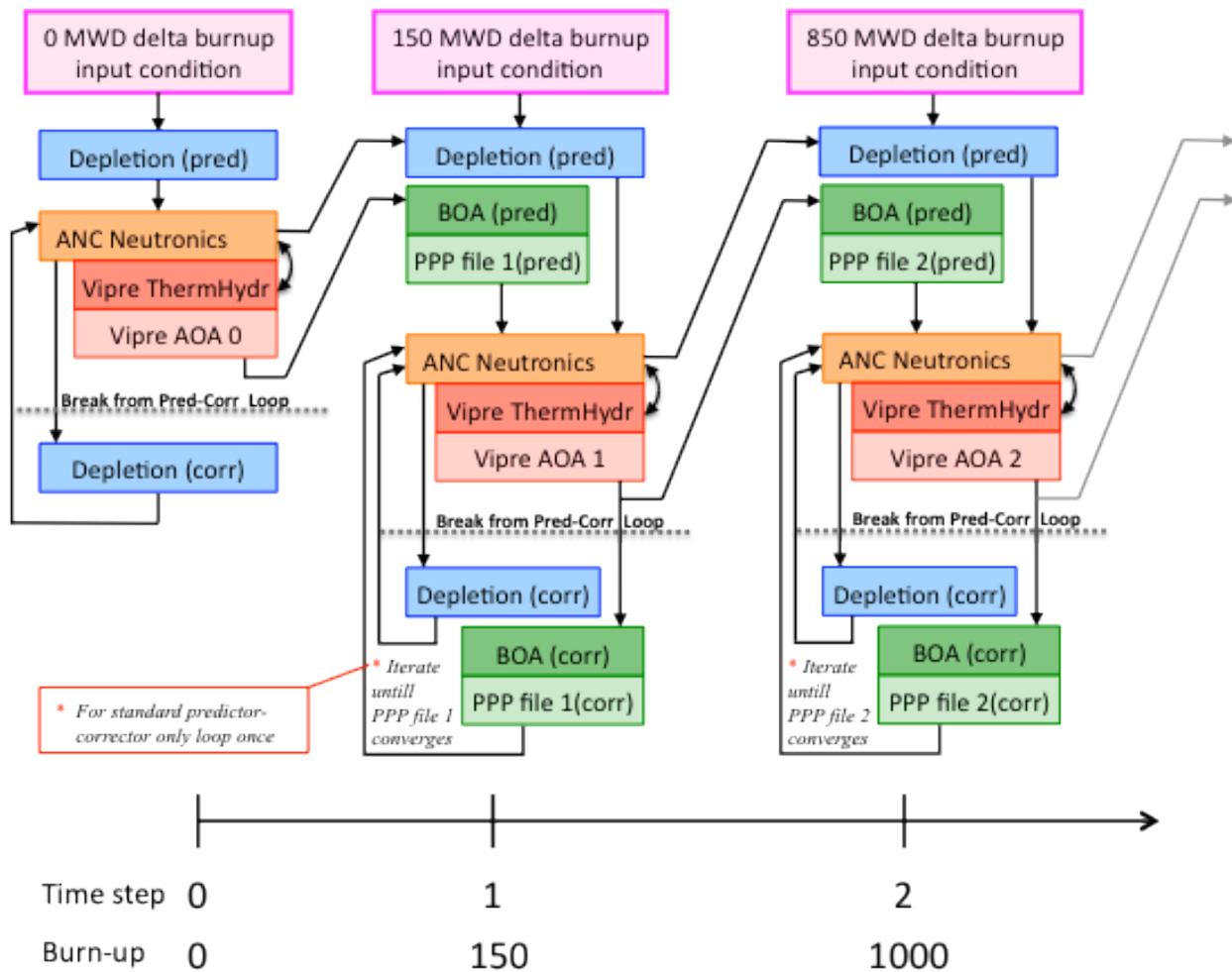


Figure 7.7: Data flow between the physics codes ANC-VIPRE-BOA for both the explicit predictor-corrector and the implicit Crank-Nicolson time-advancement methods.

Figure 7.7 is an alternate way of looking at the data flow between the physics codes that follows a sequence of steps as data is passed and solutions obtained during the coupled time integration process. It also helps to illustrate how introducing a simple iteration loop into the standard predictor-corrector algorithm leads to a “Crank-Nicolson” solution. A Crank-Nicolson time integration method is an implicit approach which can be represented here if the “starred” values in equations (5) and (6) are replaced by the actual “end-of-state” values, i.e.

$$I_{n+1} = I_n + \frac{\Delta t}{2} \left[\frac{\partial}{\partial t} (I_n, \phi_n) + \frac{\partial}{\partial t} (I_{n+1}, \phi_{n+1}) \right] \quad (7)$$

$$B_{n+1} = B_n + \int_{t_n}^{t_n + \Delta t} \left(\frac{\partial B}{\partial t} (B, (V, T, \rho)_n, (V, T, \rho)_{n+1}) \right) dt \quad (8)$$

These equations can be solved by performing a fixed point iteration over the loop shown in Figure X4 until a convergence metric is satisfied, such as the BOA PPP file converging.

7.5.3 Comparison of selected results for a Watts Bar Cycle 1 calculation

The revised ANC-VIPREW-BOA multiphysics simulation capability with improved coupling was used to model Watts Bar Unit 1 Cycles 1-3. The previous loosely coupled simulation [5] showed oscillations in the boron deposited in the CRUD as shown in Figure 7.8. This also affected the core axial offset predictions as shown in Figure 7.9. Since the BOA results from one time step were not provided to the neutronic/T-H model until the next time step, the resulting heat flux and sub-cooled boiling distributions calculated by the neutronic and T-H models were inconsistent with the boron deposition calculated by the CRUD/chemistry model at the same time step. Boron deposited in one time step would result in suppression of the neutron flux in the upper spans of the core in the next time step. That would reduce sub-cooled boiling in those regions, and reduce boron deposition in the following time step. The reduced boron deposition would then increase heat flux and sub-cooled boiling in the upper spans of the core in the next step. This would in turn increase boron deposition in the following step.

With the improved coupling, the oscillations in the model were greatly reduced when normal large depletion steps (1000-2000 MWD/MTU) were taken in the neutronics model. Some oscillations were still present as shown in Figures Figure 7.8 and Figure 7.9. This is because the CRUD/chemistry model was still depleted with finer time steps than the neutronic model. A neutronic burnup step of 2000 MWD/MTU, which is typically used for the last 2/3rds of the cycle depletion, represents a time depletion of about 52 days. The CRUD/chemistry model would use about 12 time steps over that same period, but with fixed inputs for the heat flux and T-H conditions that did not reflect the variation in boron deposition during those steps. This can be seen in Figure 7.8. The boron drops at about 310 and 360 days, which correspond to the ANC time steps. It then varies in between the 310 and 360 days steps, which represent the BOA calculations alone without direct feedback to ANC and VIPREW.

To further reduce the small oscillations now present with the improved coupling, more consistent time steps for the neutronic depletion and the CRUD/chemistry CRUD and boron deposition were used. Neutronic depletion steps of 250 MWD/MTU, or about 6.5 days were used for the fine time step calculations. As can be seen in Figure 7.8, the oscillations in boron deposition are now quite small. For these calculations, BOA is still using about 3 steps for each ANC neutronic step. The resulting small oscillations in boron deposition are not large enough to significantly affect the axial power distribution. Figure 7.9 shows much smoother behavior in axial offset late in the cycle for the models with improved coupling. The fine time steps appear to eliminate any indications of oscillations in the axial power distribution. The improved coupling has greatly reduced or eliminated the previous oscillations in axial power distribution and represents an improvement in the multi-physics modeling capability.

Note that there are two reasons CASL is expending resources on baseline proprietary software that is never intended for broad release as part of VERA.

1. Insight gained through coupling of these physics components will inform how the advanced physics components should be coupled.
2. Baseline capability can be exercised to complete sensitivity analysis and uncertainty quantification, informing resource allocation decisions regarding advanced physics components.

References for Section 7.5: ANC9.5-VIPREW-BOA

1. Liu, Y. S., S. L. Davidson, and N. A. Silva, ANC-A Westinghouse Advanced Nodal Computer Code, WCAP10965-P-A, 1998.
2. VIPRE-01 Modeling and Qualification for Pressurized Water Reactor Non-LOCA Thermal-Hydraulic Safety Analysis, WCAP-15306-NP-A, October 1999.
3. Boron-induced Offset Anomaly (BOA) Risk Assessment Tool, Version 3.0 User’s Manual, Electric Power Research Institute, November 2010.
4. R. Schmidt, “A description of the algorithmic steps and data transfer needs for coupling ANC-VIPRE-BOA in a time-consistent manner”, VRI Kanban ticket 2375, Feb., 2012.
5. J. Secker, R. Milanova, Y. Sung, P. Hilton, B. Coulter, K. Belcourt, R. Schmidt *Coupled ANC/VIPRE/BOA Multi-Physics Results for Watts Bar 1 Cycles 1-3, L1: CASL:P2:03, CASL-2011-0125-000-CI, Sept. 29, 2011.*

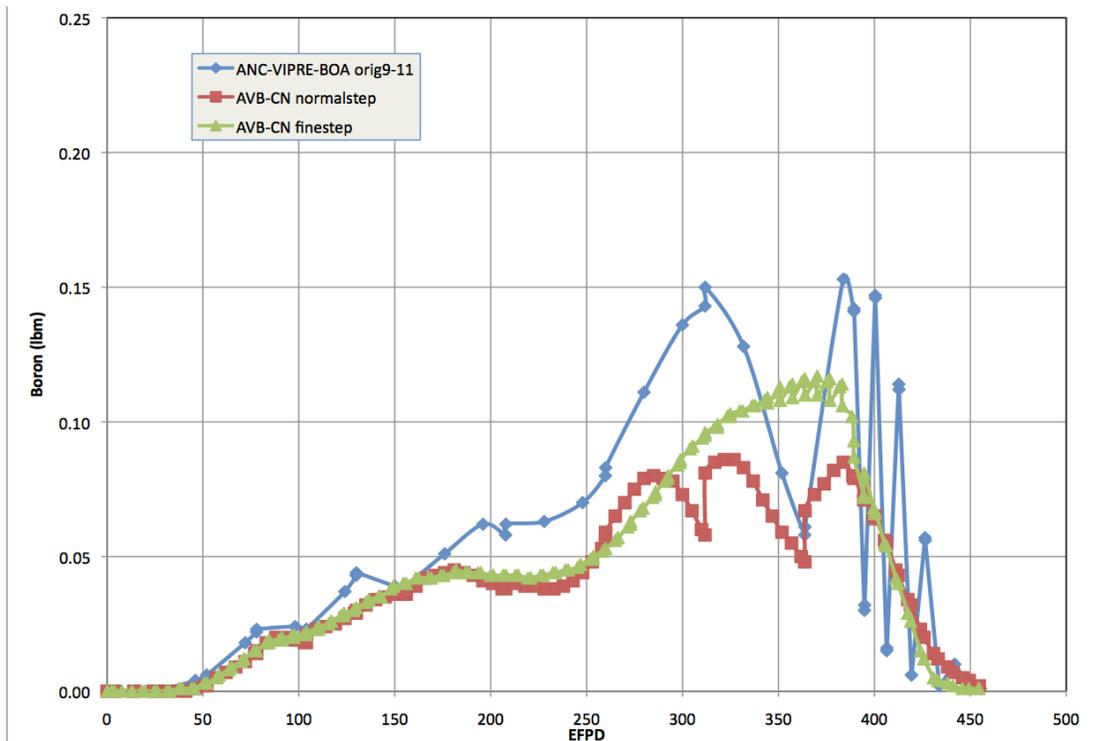


Figure 7.8: Comparison of boron mass predicted during Watts Bar Cycle 1.

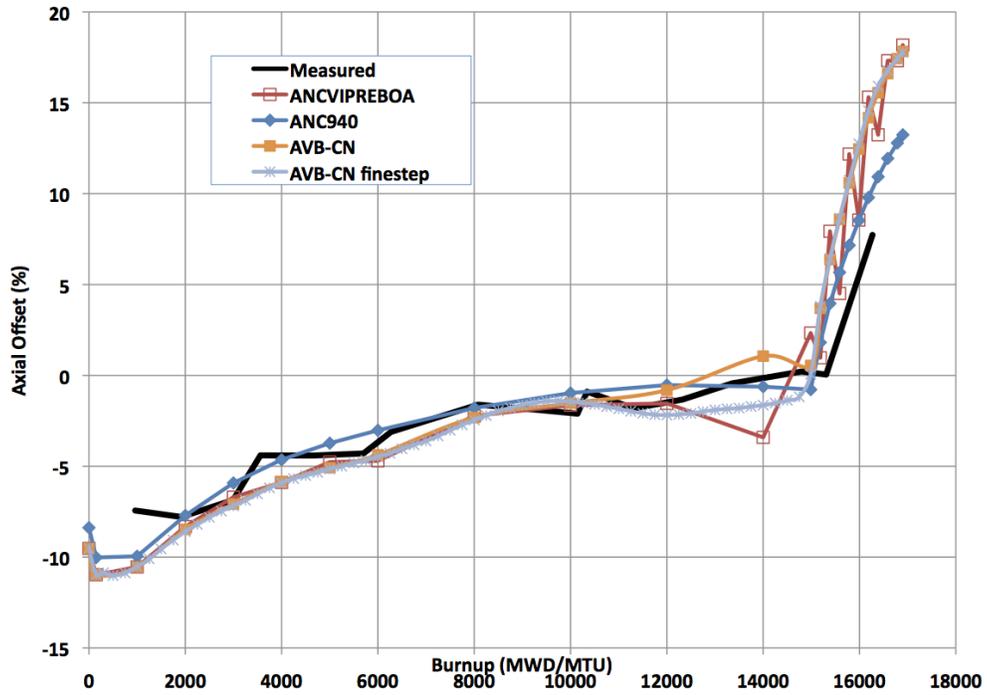


Figure 7.9: Comparison of Axial Offset predicted for Watts Bar Cycle 1.

8 VERA INITIAL AND DEMONSTRATION CAPABILITIES

8.1 DeCART

DeCART (Deterministic Core Analysis based on Ray Tracing) [1] is a whole core neutron transport code capable of direct sub-pin level flux calculation at power generating conditions of a LWR. It requires neither a priori homogenization nor group condensation as needed in conventional reactor physics calculations. DeCART solves the three-dimensional neutron transport problem employing a 2D-1D method in which the planar solution is performed using the Method of Characteristics (MOC) transport solutions and the axial solution is performed using the Nodal Expansion Method (NEM) diffusion kernel. The DeCART code has various computational capabilities to solve not only the steady state eigenvalue problems but also the transient fixed source problems. The steady state calculation can be performed in various modes which may consist of a criticality search, branch restart, and/or depletion. The depletion calculation is performed using the predictor/corrector method and has been well validated for a wide range of PWR and BWR applications. More details can be found in [2, 3].

DeCART has been deprecated in favor of the new pin-resolved neutronics capability provided by MPACT (Section 4.4).

8.2 Star-CCM+

Star-CCM+ is a commercial finite-volume-based computational fluid dynamics code and modeling package from CD-Adapco [4]. CASL has licenses to use Star-CCM+ and has placed the code into the repository for access by authorized users. The official version of Star-CCM+ in VERA 2.0 has been upgraded from 4.06 to 6.04, which includes several updates to Star-CCM+, including two-phase flow models. Details about the capabilities, numerical methods, and use of Star-CCM+ are available from CD-Adapco and not presented here.

The primary use of Star-CCM+ in VERA has been for coupling with DeCART, as described above in Section 8.3. Using the flexibility accommodated by LIME, a socket based server-client interface is used to control the Star-CCM+ solution for the coupled simulation for these simulations. In addition, a few stand-alone calculations have also been explored for potential insight and application in the future.

8.3 DeCART-Star-CCM+

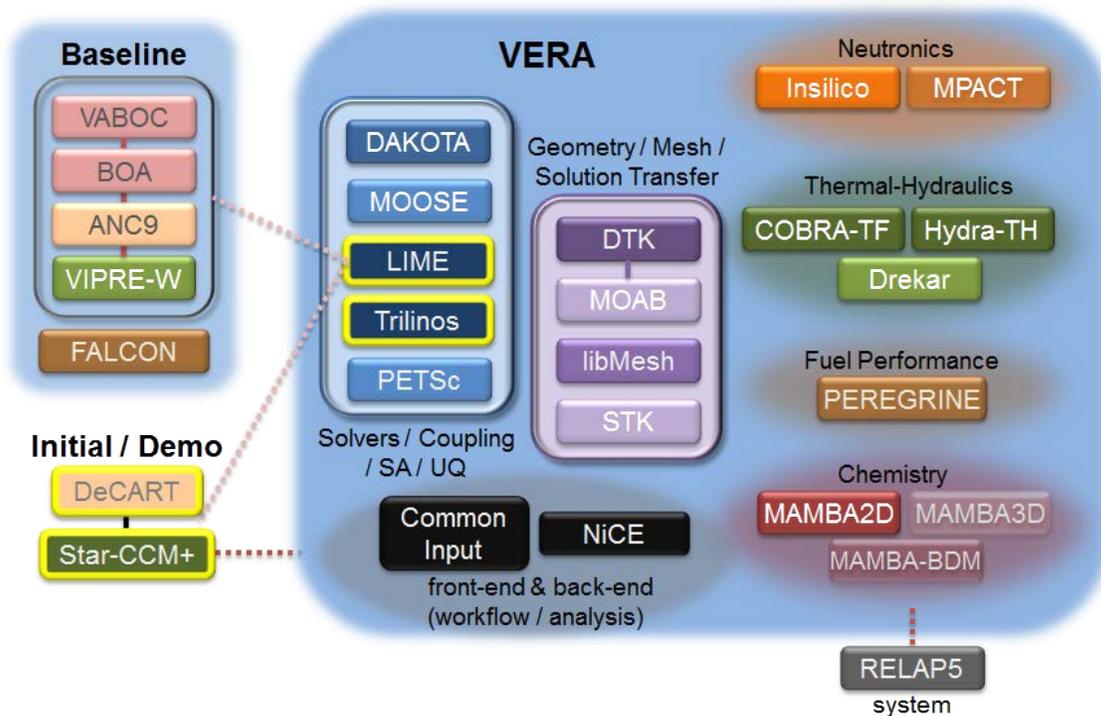


Figure 8.1: Components used in DeCART-Star-CCM+ capability.

The DeCART–Star-CCM+ coupled multiphysics capability has been significantly improved since the release of VERA 1.0. However, most of these improved capabilities were incorporated into the VERA 1.1 [5] or VERA 1.2 [6] interim releases. These changes have included the development of a LIME-based multiphysics driver, many improvements to the model evaluators for both codes and, work on developing parallel file I/O for data transfers that has resulted in major execution time improvements. A summary of this DeCART-Star-CCM+ capability as it exists in VERA is provided here.

The VERA Release 1.0 report [7] outlines the development of the socket based server-client interface which is used to control the Star-CCM+ solution. The data transfers between DeCART and Star-CCM+ are handled using serial file I/O. The report also outlines the future of the data transfer. An intermediate step is to implement user coding inside Star-CCM+ to perform the file I/O in parallel. DeCART has the ability to read and write multiple files based on the domain decomposition used in Star-CCM+. The user coding is created to emulate the same functionality as the tables used in the previous version only in parallel.

The user coding used by Star-CCM+ consists of routines written either in Fortran or C/C++. This code is then compiled into a dynamically linked library. At run time, Star-CCM+ is capable of loading this library and calling its member functions. The only additional step that must be taken when building the user code library is to write a simple library registration function so that Star-CCM+ knows what routines are available. The user coded routines can be used by Star-CCM+ in some places directly (e.g. like the definition of the energy source) or as scalar or vector field functions. The field functions differ in that they can be utilized almost anywhere within

STAR-CCM+ (e.g. they can be plotted or define nearly any user definable property). In addition to the user coding routines and field functions made available by the library, the "reports" feature in STAR-CCM+ is used to assist in controlling the execution of user coded routines. This ability is especially important for the initialization.

Along with the addition of user coding the java interface is also updated to automate the set up and calling of the user coding. Backwards compatibility for using the original table methods is also maintained if Star-CCM+ is running in serial. These updates are included in 'LimeClientME_v2.java' inside the StarCCMClient repository.

In addition to adding the user coding to perform the parallel file I/O, the functionality to call Star-CCM+ in parallel is also needed. The feature to run Star-CCM+ in parallel does not require any modifications to Star-CCM+ but instead the driver is modified to accept the command line options. The 'staropt' command line option was added to the driver to pass additional options for all of the parallel commands to Star-CCM+.

The final addition to the coupled code system is to allow both codes to be run in parallel using parallel file I/O for data transfers. In order to do this, the driver code is modified to send the MPI communication environment to both DeCART and Star-CCM+. Since Star-CCM+ controls its own parallel communication as a separate executing process, all of the calls in the Star-CCM+ model evaluator must only be called by the master process.

In order to test the user coding and assess the improvements running the coupled code system in parallel a 3 by 3 pin cell problem, shown in Figure 8.2, is run.

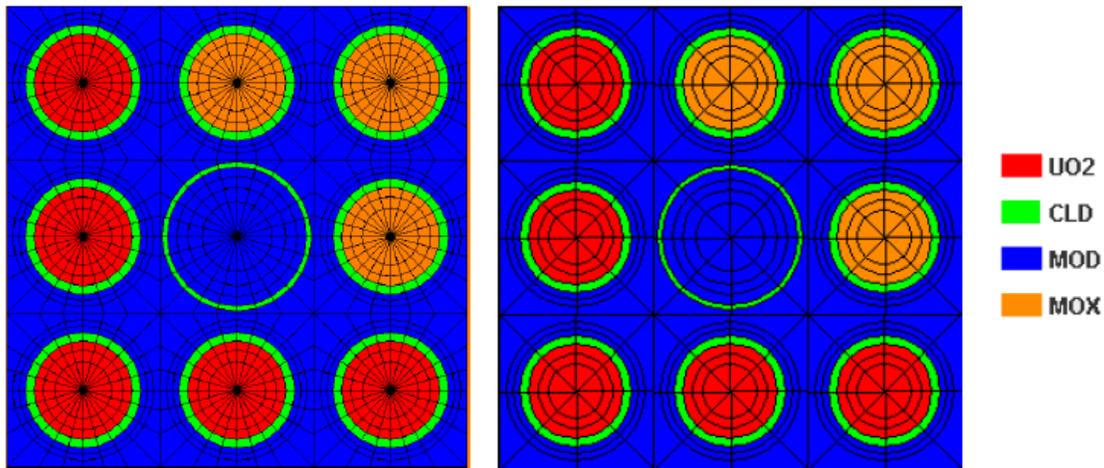


Figure 8.2: Star-CCM+ (Left) and DeCART (Right) mesh for demonstration problem.

The 3 by 3 problem is described in detail in reference [8]. The same case is run with 16 processors for both DeCART and Star-CCM+ on 'u233.ornl.gov'. The eigenvalue, peak pin power, and average fuel temperature are compared for the serial and parallel cases. Table 8.1 shows the comparison of the key solution values and the run times. The comparison of the solution shows very good agreement and the increase in speed shows significant gains. Continuing work should continue to improve the scaling of the coupled codes.

Table 8.1: Comparison of DeCART-Star-CCM+ serial and parallel solution and run times.

	k-effective	Peak Pin Power	Average Fuel Temperature	Total Run Time	DeCART		Star-CCM
					Computation (s)	File I/O (s)	Total Time (s)
VERA 1.0	1.271750	1.6805	408.10	600.570	37.31	339.920	223.340
VERA 2.0	1.271775	1.6850	408.08	98.665	1.35	37.103	60.212

8.4 Insilico-Star-CCM+

Following the refinement of DeCART-Star-CCM+ coupling described in Section 8.3, the framework developed was used to demonstrate coupling of the Denovo-based neutronics capability now known as Insilico and Star-CCM+. This demonstration was successful, but has not been developed further, and is not under continuous or nightly testing to ensure correctness. If this capability were to be needed further, it could certainly be used, but would require development.

References for Section 8:

1. H.G. Joo, et al., "Dynamic Implementation of the Equivalence Theory in the Heterogeneous Whole Core Transport Calculation," Proc. Int. Conf. New Frontiers of Nuclear Technology: Reactor Physics, Safety and High-Performance Computing (PHYSOR 2002), Seoul, Korea, October 7-10, 2002, CD-ROM, American Nuclear Society, 2002.
2. Hursin, M., Kochunas, B., Downar, T., DeCART v2.05 User's Manual. School of Nuclear Engineering and Radiological Sciences, University of Michigan, 2009
3. Hursin, M. Full Core, Heterogeneous, Time Dependent Neutron Transport Calculations with the 3D Code DeCART, PhD Dissertation, University of California, Berkeley, 2010
4. http://www.cd-adapco.com/products/star_ccm_plus/index.htm
5. Release 1.1 of the Virtual Environment for Reactor Analysis (VERA), CASL Level 2 Milestone Report L2:VRI.P3.01, March 31, 2011
6. Interim Release 1.2 of the Virtual Environment for Reactor Analysis (VERA), CASL Level 3 Milestone Report L3:VRI.PSS.P2.03, June 20, 2011
7. Release 1.0 of the Virtual Environment for Reactor Analysis (VERA), CASL Level 2 Milestone Report L2:VRI.1.02, March 30, 2011
8. Brendan Kochunas, "Initial Demonstration of Tightly Coupled Fluid Flow and Transport with DeCART and STAR-CCM+", CASL L2:VRI.P2.03 Milestone Supporting Documentation Report, June 30, 2011

9 CONTRIBUTING STAFF

Staff contributing to the completion of this milestone included the following individuals. Note that numerous CASL partner institutions are represented.

- Bartlett, Ross (ORNL)
- Belcourt Noel (SNL)
- Christon, Mark (LANL)
- Clarno, Kevin (ORNL)
- Collins, Ben (U. of Mich.)
- Davidson, Greg (ORNL)
- Downar, Tom (U. of Mich.)
- Evans, Tom (ORNL)
- Hamilton, Steven (ORNL)
- Godfrey, Andrew (ORNL)
- Hilton, Pete (WEC)
- Hooper , Russ (SNL)
- Jarrell, Josh (ORNL)
- Kochunas, Brendan (U. of Mich.)
- Lefebvre, Jordan (ORNL)
- Lefebvre, Rob (ORNL)
- Palmtag, Scott (Core Physics)
- Pawlowski, Roger (SNL)
- Sankaran, Ramanan (ORNL)
- Schmidt, Rod (SNL)
- Secker, Jeff (WEC)
- Simunovic, Srdjan (ORNL)
- Slattery, Stuart (U. of Wisc.)
- Sung, Yixing (WEC)
- Summers, Randy (SNL)
- Turner, John (ORNL)
- Williams, Mark (ORNL)
- Wilson, Paul (U. of Wisc.)
- Zhang, Baocheng (WEC)

Appendix A. VERA Input Format (ASCII)

An example VERA input deck, in this case for AMA Benchmark Problem 6, Hot Full-Power Beginning of Life Assembly, is shown below.

```
[CASEID]
  title 'CASL Problem 6a'
=====
! Sample input for Problem 6 (Single-assembly with T/H feedback)
!
! Draft 1 - 9/28/2012 - starting with Problem 3 input deck
!   * changing power to 100%
!   * turn on T/H feedback
!   * remove "tfuel" and "modden" because these will be set by T/H feedback
!
! Draft 2 - 10/13/2012
!   * added "mat" card in front of material cards
!
! Update 11/19/2012
!   * added explicit lattice models for:
!     * lower_nozzle_gap_height
!     * lower_pincap_height
!     * upper_nozzle_gap_height
!     * upper_pincap_height
!     * upper_plenum_height
!
! To-Do:
!   * add percent of theoretical density to pellets
!   * add grid spacer loss coefficients
!
! To process:
!   ./react2xml.pl --xml --xslt Sample_p6.inp Sample_p6.xml
=====

[STATE]
power 100.0           ! %
tinlet 620.33         ! F - 600K
boron 1300            ! ppmB
pressure 2250         ! psia

! tfuel 600.0         ! K - 600K  Not used with T/H feedback!
! modden 0.743       ! g/cc    Not used with T/H feedback!

feedback on

[CORE]
size 1                ! 1x1 single-assembly
rated 17.67  0.5      ! MW, Mlbs/hr
apitch 21.5
height 406.328

core_shape
  1
```

assm_map
A1

lower_plate ss 5.0 0.5 ! mat, thickness, vol frac
upper_plate ss 7.6 0.5 ! mat, thickness, vol frac

bc_rad reflecting

! Old material format
!
! he he 0.000176
! inc inc 8.19
! ss ss 8.0
! zirc zirc 6.56
! aic aic 10.20
! pyrex pyrex 2.23
! b4c b4c 6.56

mat he 0.000176
mat inc 8.19
mat ss 8.0
mat zirc 6.56
mat aic 10.20
mat pyrex 2.23
mat b4c 6.56

[ASSEMBLY]

title "Westinghouse 17x17"
npin 17
ppitch 1.260

! uo2 U31 10.257 / 3.1
fuel U31 10.257 95.0 / 3.1

!=== material label, key_name, density (lib_name defaults to key_name)

mat he 0.000176
mat inc 8.19
mat ss 8.0
mat zirc 6.56

cell 1 0.4096 0.418 0.475 / U31 he zirc
cell 100 0.561 0.602 / mod zirc ! guide tube
cell 200 0.561 0.602 / mod zirc ! instrument tube
cell 7 0.418 0.475 / mod mod ! empty location
cell 8 0.418 0.475 / he zirc ! plenum
cell 9 0.475 / zirc ! pincap

lattice FUEL1

200
1 1
1 1 1
100 1 1 100
1 1 1 1 1
1 1 1 1 1 100
100 1 1 100 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1

```

lattice LGAP1
200
  7 7
  7 7 7
100 7 7 100
  7 7 7   7 7
  7 7 7   7 7 100
100 7 7 100 7   7 7
  7 7 7   7 7   7 7 7
  7 7 7   7 7   7 7 7 7

```

```

lattice PLEN1
200
  8 8
  8 8 8
100 8 8 100
  8 8 8   8 8
  8 8 8   8 8 100
100 8 8 100 8   8 8
  8 8 8   8 8   8 8 8
  8 8 8   8 8   8 8 8 8

```

```

lattice PCAP1
200
  9 9
  9 9 9
100 9 9 100
  9 9 9   9 9
  9 9 9   9 9 100
100 9 9 100 9   9 9
  9 9 9   9 9   9 9 9
  9 9 9   9 9   9 9 9 9

```

```

axial A1      6.050
  LGAP1 10.281
  PCAP1 11.951
  FUEL1 377.711
  PLEN1 393.711
  PCAP1 395.381
  LGAP1 397.501

```

```

grid END inc 1017 3.866
grid MID zirc 875 3.810

```

```

grid_axial
  END 13.884
  MID 75.2
  MID 127.4
  MID 179.6
  MID 231.8
  MID 284.0
  MID 336.2
  END 388.2

```

```

lower_nozzle ss 6.05 6250.0 ! mat, height, mass (g)
upper_nozzle ss 8.827 6250.0 ! mat, height, mass (g)

```

```
!! dancoff                ! assembly_dancoff_map
!! 0.000
!! 0.287 0.315
!! 0.287 0.315 0.315
!! 0.000 0.287 0.286 0.000
!! 0.287 0.316 0.316 0.284 0.299
!! 0.288 0.317 0.316 0.287 0.267 0.000
!! 0.000 0.286 0.286 0.000 0.270 0.286 0.321
!! 0.287 0.319 0.319 0.286 0.315 0.335 0.333 0.337
!! 0.323 0.322 0.321 0.322 0.320 0.322 0.323 0.322 0.310
```

[EDITS]

! 3in intervals in active fuel

```
axial_edit_bounds
11.951
15.817
24.028
32.239
40.45
48.662
56.873
65.084
73.295
77.105
85.17
93.235
101.3
109.365
117.43
125.495
129.305
137.37
145.435
153.5
161.565
169.63
177.695
181.505
189.57
197.635
205.7
213.765
221.83
229.895
233.705
241.77
249.835
257.9
265.965
274.03
282.095
285.905
293.97
302.035
310.1
```

318.165
326.23
334.295
338.105
346.0262
353.9474
361.8686
369.7898
377.711

[INSILICO]

```
eq_set      ld
eigen_solver  arnoldi
tolerance    1e-06
Pn_order     1
dimension    3
mesh         4
!  eigenvalue_db:
      k_tolerance  0.0001
      L2_tolerance 0.001
      energy_dep_ev false
mat_library  test_comp.sh5
max_delta_z  2.54
num_blocks_i 40
num_blocks_j 40
num_z_blocks 64
num_groups   23
num_sets     6
!  quadrature_db:
      azimuthals_octant  4
      polars_octant      4
      quad_type          qr
!  silo_db:
      silo_output vera
!  upscatter_db:
      upscatter_tolerance 1e-05
xs_library  "v7-238ir"
new_grp_bounds
      8.2085e+05
      1.1109e+05
      5.5308e+03
      1.8644e+02
      3.7612e+01
      3.5379e+01
      2.7697e+01
      2.1684e+01
      2.0397e+01
      1.5968e+01
      7.1500e+00
      6.7000e+00
      6.3000e+00
      1.0970e+00
      1.0450e+00
      9.5000e-01
      3.5000e-01
      2.0600e-01
      1.0700e-01
```

5.8000e-02
2.5000e-02
1.0000e-02
1.0000e-05

[COBRATF]

nfuel 3
irfc 1
epso 0.001
oitmax 3
iitmax 40
gridloss END 0.9070
gridloss MID 0.9065

Appendix B. VERA Input Format (Parameters)

Detailed documentation of the VERA Common Input is available on the CASL “wiki”, CASLpedia, which uses the same software (MediaWiki, <http://www.mediawiki.org/>) as the well-known Internet Encyclopedia, Wikipedia (<http://en.wikipedia.org/>).

Although the formatting of the content for a PDF document is significantly less attractive since it is meant to be viewed on-line (in a web browser), it is available on request.

Appendix C. DataTransferKit Reference

Documentation of the DataTransferKit component is available separately.

Appendix D. Insilico Reference

Documentation of the Insilico neutronics component is available separately.