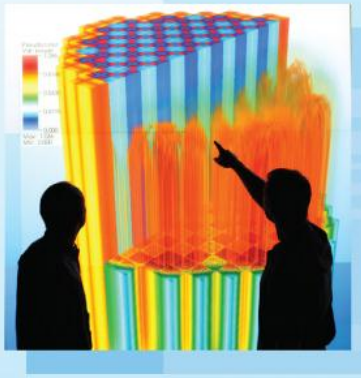
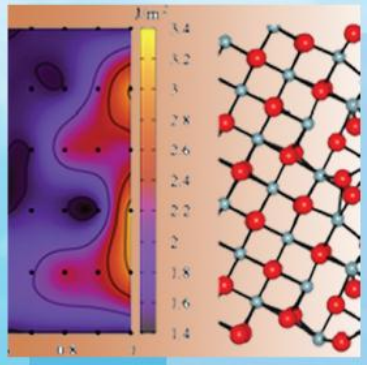
CASL-U-2011-0169-000

Power uprates
and plant life extension



Engineering design
and analysis

Science-enabling
high performance
computing



Fundamental science

Plant operational data

Consortium for Advanced Simulation of LWRs

L3:VRI.VERA.P3.02

Chris Baker
ORNL

Completed: 9/11/11

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

Technical Note

Virtual Reactor Integration

VERA

From: Chris Baker, Ross Bartlet, Kevin Clarno, Ben Collins, Tom Evans, Roger Pawlowski, Rod Schmidt, John Turner

Number: VRI-11-002

Date: September 11, 2011

Subject: Domain Model Specification for VERA: Version 1.0 (September 2011) (Rev. 1)

Executive Summary

We present a high-level domain model specification and scoping for the Virtual Environment for Reactor Analysis (VERA) for the CASL project. This model defines a number of VERA subdomains and establishes a ubiquitous langauge for VERA that will aid in communication across CASL's developers and stakeholders and will help to prioritize and drive development efforts.

Contents

1 Introduction **2**

2 VERA Subdomains **2**

2.1 User GUI/Workflow/Control . . . 4
2.2 Reactor-Aware FACADEs . . . 4
2.3 Simulation Workflow Control Scripting . . . 4
2.4 Physics Models . . . 4
2.4.1 VERA-Supported Physics Models . . . 5
2.4.2 External Physics Models . . . 6
2.5 Visualization . . . 6
2.6 Common Reactor Input Specification . . . 6
2.7 Geometry Specification . . . 6
2.8 Mesh Generation . . . 7
2.8.1 Basic Automatic Mesh Generation . . . 7
2.8.2 Embedded Automatic Mesh Generation . . . 7
2.8.3 Advanced Mesh Generation . . . 8
2.9 Materials Interface . . . 8
2.10 Data Mapping . . . 9
2.11 Simulation Post-Processing . . . 9
2.12 Multiphysics Coupling . . . 10
2.13 Basic Numerical Algorithms . . . 10
2.14 Verification and Uncertainty Quantification (VUQ) . . . 11

3 VERA Base **11**

4 Other Architectures Considered **12**

5 Conclusions and Future Work **12**

1 Introduction

This document presents a Domain Model (DM) [8, 5] that spans the entire VERA problem space. The objective of this domain model[1] is to provide clearly defined domain definitions and create a ubiquitous language [5] for VERA. The VERA domain model is comprised of a set of subdomains, the details of which will be described in separate documents (for example, see [6] and [9]).

In addition, specific use cases for VERA are also being developed and documented separately. These will most directly impact usability layers such as the *Reactor-Aware FACADEs* described in Section 2.2 and the *User GUI/Workflow/Control* layer in Section 2.1.

The proposed domain model for VERA along with its subdomains are illustrated in Figure 1. This domain model diagram is expressed as a UML [8] package diagram. Subpackages are used to represent the VERA subdomains which might be loosely correlated with actual collections of software and the relationships in the diagram represent dependency relations (and not objects that point to each other). All of the subdomains shown inside of the "VERA" enclosure package are considered to be in the scope of CASL VERA development while those outside of the enclosure would be considered to be outside of the scope of VERA but could be leveraged by VERA. In fact, CASL could even contribute to the development of these out-of-scope areas but they would not be considered to be core development efforts for CASL (or at least not by the VRI focus area). A subset of the VERA subdomains is also shown called *VERA Base* which represents the set of VERA software that would be open source and have open development and/or releases (see Section 3).

The primary audience for this document are the CASL developers that will be constructing VERA software. The categorizations and terminology expressed in this document will be used by VERA developers to aid in communication and help plan and manage the development efforts. A secondary audience are VERA stakeholders, such as potential users and testers of VERA. For VERA stakeholders, the definitions and the language defined in the VERA Architecture model will aid in communication with the VERA developers and help in the planning an prioritization of feature sets.

2 VERA Subdomains

Here we describe the various subdomains that are related to VERA. The purposes for this categorization are to

- define clear nomenclature (i.e. a Ubiquitous Language) for future discussions about VERA,
- partition and prioritize the work related to creating VERA, and
- define what is and what is not within the scope of CASL and VRI related to the development and use of VERA.

The subdomains are approximately ordered in subsequent sections starting with those closest to the user and progressing to lower-level infrastructure subdomains primarily of interest to VERA software developers and integrators. Because of this ordering, terms mentioned in one subdomain section may not be defined until later subdomain sections (i.e. because the software dependencies generally go in the opposite order).

[1] A domain model is comprised of a set of mental concepts and relationships that are used to create a simplification of some problem or domain. The goal of Domain Driver Design [5] is to try to make the software match the domain model as close as possible so that the two evolve together as the software is developed.

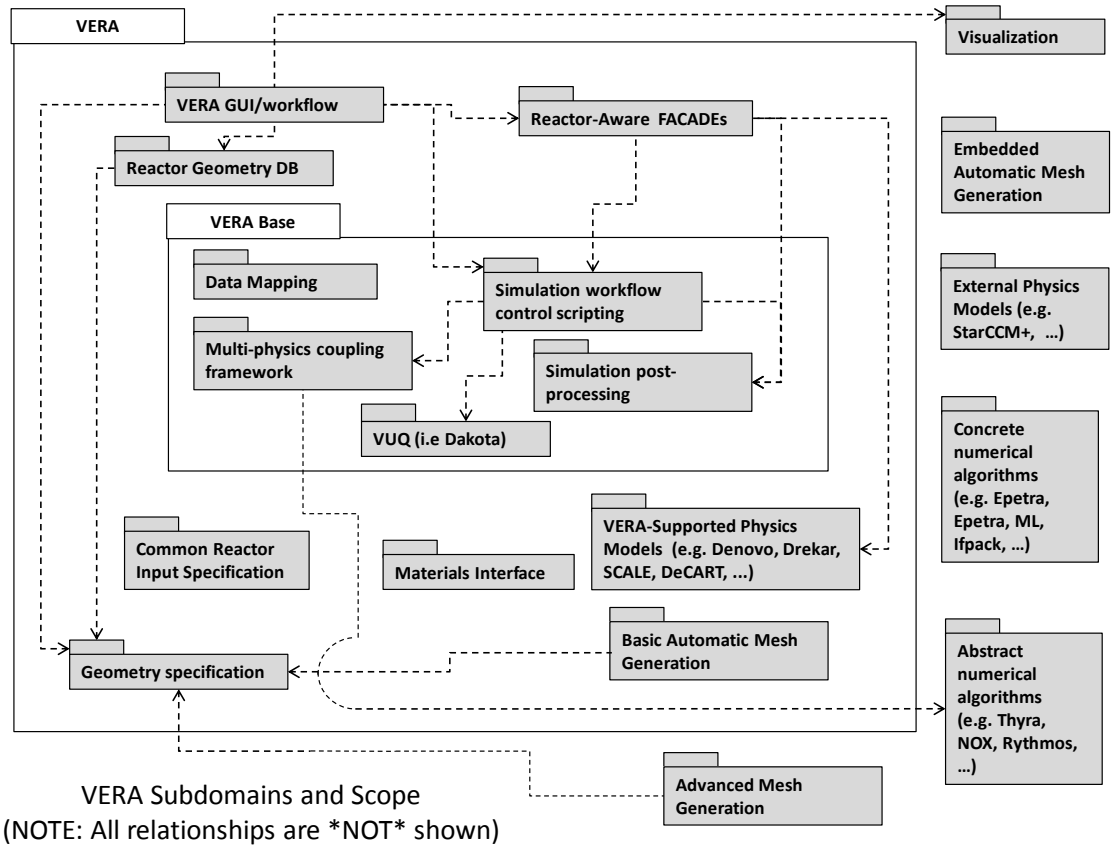


Figure 1: VERA Domain Model. (UML Package Diagram.)

2.1 User GUI/Workflow/Control

The outermost layer in VERA would be the *User GUI/Workflow/Control* layer which is how non-programming savvy users would use access VERA simulation capabilities. Development of this layer will evolve through iterative feedback with customers represented by AMA FA staff and other CASL-friendly individuals.

2.2 Reactor-Aware FACADEs

Sitting on top of a general the simulation workflow control scripting layer (see Section 2.3) will be a set of specific reactor-aware simulator FACADEs[2]. It is at this layer that VERA capability equivalent to what in common industry parlance would be called a "core simulator" would reside. In addition, other simulators could be developed as needed to drive more specialized types of reactor analysis. These reactor-aware FACADEs would be thin layers sitting on top of the simulator workflow control scripting layer described in Section 2.3. Given that this layer would be built on this general scripting layer, it would also be script-able and customizable so as to allow the creation of higher-level simulation and analysis workflows in a recursive nature. For example, the VERA basic core simulator would be represented as a script-able component in the scripting layer and therefore could be used to create other simulation analysis workflows (e.g. parameter sensitivity, UQ analysis, etc.).

Related to a reactor-aware FACADE would be a set of (test) simulation scenarios comprised of all the input files – specified in the "common reactor input specification" (Section 2.6) and "geometry specification" (Section 2.7) – needed to run various specific simulation scenarios.

An example of a Reactor-Aware FACADE would be a LIME-coupled ANC/VIPRE/BOA simulation object that wraps the mixed-language compiled C/C++/Fortran executable along with any necessary input and output processing tools needed to allow power-users to drive simulations using the simulation workflow control scripting layer. Packaged along with the ANC/VIPRE/BOA FACADE would be as set of input files given in standard reactor input specification and geometry specification formats for running various scenarios. Automated tests would be defined that would invoke the ANC/VIPRE/BOA FACADE for the given set of scenarios and would compare the output to accepted results. These tests would serve as both regression tests and as a kind of infallible tested documentation for how to setup and run the reactor simulations using the FACADE.

2.3 Simulation Workflow Control Scripting

A *Simulation Workflow Control Scripting* layer will be acquired or developed for VERA that will provide the most basic power-user level interface to VERA. Using a high-level scripting language (e.g. Python), scripting-savvy power users of VERA will be able to access multiple pre-packaged VERA Reactor-Aware FACADEs (for single and multiphysics simulation codes) and be able to create different sequences of individual simulation calculations including pre- and post-processing. This layer will provide the most basic mechanisms to create repeatable simulations. The simulation workflow control scripting layer will likely interface with a version control repository to manage the versions of input and output files and provide the most basic foundation for simulation provenance and reproducibility. (This type of provenance and reproducibility is a critical element in accumulating V&V evidence needed to establish the credibility of a simulation code.)

The building blocks exposed within this scripting layer will be the Reactor-Aware FACADS as well as several other types of tools including, for example, VUQ analysis (e.g. DAKOTA), external post-processing, and visualization. Examples of this type of software include Salome, iSight, Swift, and VisTrails.

2.4 Physics Models

Physics Models are incorporated into VERA using the *Multiphysics Coupling Framework* (see Section 2.12). Each physics model is a domain-specific application component; examples include neutronics, thermohy-

[2]The term FACADE is an accepted software Design Pattern [7] that used to denote a design where one creates a high-level user friendly wrapper that hides implementation details of a more complex set of software.

draulics, chemistry, materials, and other areas. Within each physics module there can be additional sub-components. For example, neutronics includes multigroup cross section generation, transport, and depletion [6]. A central purpose of VERA is to leverage domain expertise, expressed through individual physics components, by using the *Multiphysics Coupling Framework* and *Basic Numerical Algorithms* to couple the domain-specific components into a multidisciplinary application.

Physics models can be classified in two categories: those directly supported by VERA developers and those that are externally supported. Directly supported physics models are those that are explicitly targeted by other VERA components (Mesh Generation, FACADES, etc) and are required to solve CASL challenge problems and will not be supported by CASL development funds. External physics models includes those areas of physics that may be required to solve future problems targeted by VERA users or sponsors, but these components are not immediately required for CASL challenge problems.

Note that third-party codes developed and maintained outside of CASL are not necessarily treated as external physics models in VERA. For example, COBRA, an existing sub-channel thermalhydraulics code package, may be targeted by VERA for sub-channel physics. This would be considered a VERA-supported physics model even though it is an external code. The distinction lies in the acceptance and support by CASL and VERA developers, not in the origination of the software.

Each physics model should provide a domain-specific design/requirements specification that describes the coupling input and output. An example is the Neutronics specification listed in [6].

2.4.1 VERA-Supported Physics Models

The types of Physics Models that will likely be directly supported by VERA include:

- **Chemistry**
  - CRUD formation
  - corrosion
- **Fuel Performance**
  - thermo-mechanics
  - materials
- **Neutronics**
  - cross-section generation
  - transport
  - depletion
- **Thermal Hydraulics**
  - sub-channel
  - CFD
  - conjugate heat transfer
- **Structural Dynamics**

The actual set of physics models needed to address VERA milestones will be developed and maintained in a separate document. Each general physics model can include specific models that are geared to solving CASL challenge problems.

2.4.2 External Physics Models

External physics models include those domain areas that are not directly targeted by VERA components needed to solve CASL challenge problems. This list could include nearly any area of physics and will evolve over time.

2.5 Visualization

Visualization involves the rendering of complex 2D and 3D spacially related simulation fields on various geometries. This type of software must be able to handle massive amounts of parallel simulation data and therefore must be aware of the parallel machine (and must run on the parallel machine in many cases) and must be mesh aware. However, such tools are usually not aware of the details of the discretizations of the various physics models.

Visualization is a critical aspect of modeling and simulation of physical systems and much software exists this area, for example ParaView and VisIt. However, VERA will not provide any official support for specific visualization tools. Users of VERA will be able to plug in whatever visualization tools they want into a VERA simulation or analysis workflow that supports the given mesh/data formats.

2.6 Common Reactor Input Specification

Common reactor input specification refers to a single collection of declarative input files that contain of all of the input parameters, initial conditions, physics equations, and other data that are necessary to define a VERA (multiphysics) simulation problem. These files will contain no duplication of information and may be linked together (e.g. with include statements). The specification in this single set of input files will be mapped to the inputs to the specific individual physics simulation codes in any given VERA multiphysics simulator. There will be no duplication of information that would otherwise result in errors, maintenance problems, and increased difficulty of use. The geometric specification described in Section 2.7 would either be pointed to in these input files or could be directly embedded in the input files.

Note that having a single unique specification for input parameters facilitates various VUQ studies that need to systematically change and vary parameters over multiple simulation runs.

A strong candidate for the specification language would be XML with the addition of other tools to help in the manipulation of the files. For example, the Trilinos class Teuchos::ParameterList might be a reasonable intermediate representation for such an input specification since it supports serialization to and from XML and there exists emerging supporting tools for user manipulation (e.g. the Trilinos package Optika).

2.7 Geometry Specification

The *Geometry Specification* subdomain refers to the creation of a single declarative specification of all of the geometric information needed to define a nuclear reactor core simulation, challenge problem simulation, or subset of the domain for testing. This should cover the entire domain pellet, pin, spacer, assembly, core, lower and upper plenum, (all elements of the primary system which is the domain of CASL but not necessarily the secondary system which is outside of the domain of CASL). It is imperative that the geometry be defined in a way that allows for the use of multiple codes with multiple use cases that may cover a wide variety of geometric resolutions. For example, core follow and CIPS require the primary system, core, and assembly definitions, but the specific resolution of pellets and grid spacers may not be required; GTRF and PCI will require a geometry that resolves the individual pellets, grid spacer, and mixing vanes, but may not requires the full primary system.

Candidates for the foundation for the gemonetry specification include SCALE and MeshKit.

A few specific aspects that need to be consided in relation to the geometry specification include tags, file formats, a geometry database, and mesh integration.

The geometry specification will require tags to identify volume IDs, boundary IDs, and material IDs for each volume. These tags are required to facilitate certain types of physics models, data mapping across different meshes, and handling of in-isotropic materials.

The geometric specification will be contained in a hierarchy of files that allow for the viewing and manipulation by users and scripting (e.g. XML) for uncertainty quantification or batch processing. (Note that having a single unique specification for geometric specification facilitates various VUQ studies that need to systematically change and vary geometric shape parameters over multiple simulation runs.) The files will be hierarchical (allowing for the inclusion of other files) to simplify, and allow for, the inclusion of multiple resolutions.

A *Reactor Geometry DB* of common nuclear reactor geometries would be maintained to allow users of the *Reactor-Aware FACADEs* of VERA to quickly assemble the needed inputs to simulate common nuclear reactors. This database will include all resolutions of the geometry with various modern PWR designs. Simplifications to the geometry definition (e.g. quarter symmetry) will be handled by the *Reactor-Aware FACADEs* and will not be contained in the*Geometry Specification*. In addition, there will be an *Assembly Geometry DB* of specific fuel assemblies that have been reviewed and approved for accuracy.

The *Geometry Specification* is intended to be a single definition of the geometric components that would not be duplicated in any way in any of the inputs to a VERA simulation. However, because of the *Mesh Generation* challenges, there is no means of guaranteeing all geometry is consistent for all codes and problems. The *Reactor-Aware FACADEs* will ensure that there is consistency between mesh and geometry for *Embedded Automatic Mesh Generation* and *Basic Automatic Mesh Generation*, but not *Advanced Mesh Generation*.

2.8 Mesh Generation

The subdomain of *Mesh Generation* refers to the software needed to create an appropriate triangulation of a 1D, 2D or 3D region of space given a complete geometric specification. Here we partition mesh generation into three separate subdomains: basic automatic mesh generation, embedded automatic mesh generation, and advanced mesh generation.

2.8.1 Basic Automatic Mesh Generation

The subdomain of *Basic Automatic Mesh Generation* refers to the software involved in the automatic generation or regeneration of specialized 1D, 2D, or simple 3D meshes in a basic nuclear core simulator. Because of the restricted nature of the target physics models, automated methods can be (and have been) developed for a number of codes. These meshing methods are needed to create a basic nuclear core simulator that allows for various geometric changes that are induced by end users (i.e. that have no knowledge or expertise in meshing or the specific discretization methods).

Because such methods are central to any user-friendly basic core simulator, support for such meshing methods will be supported by VERA. A key requirement is that they be accessible by common scripting tools so that they may be invoked by the *Reactor-Aware FACADEs*.

One example of a tool that could be leveraged to develop a basic automatic mesh generation capability is CUBIT.

Note that one type of pseudo mesh generation that can be automated in some cases, even for fairly sophisticated 3D unstructured meshes, is mesh refinement. Automatic mesh refinement is critical in mesh convergence studies. Tools exist that can aid in such implementations such as DAKOTA/Percept. To the extent possible, VERA will support basic mesh refinement software.

2.8.2 Embedded Automatic Mesh Generation

The subdomain of *Embedded Automatic Mesh Generation* refers to automatic mesh generation capability embedded within individual existing *Physics Models*. These methods are generally the same as described for basic automatic mesh generation except they are embedded in existing physics codes are are not reusable in other physics codes. Examples of existing physics codes which include embedded automatic mesh generation

are DeCART, BOA, and Denovo. These tools are complete and sufficient for their applications but may have limitations (simplified geometries, smearing materials, etc.) that make them insufficient for use by other *Physics Models*.

These tools will not be maintained or developed by VERA developers but instead by the groups maintaining the specific physics codes. However, the *Geometry Specification* will define the geometry that is input to the specific *Physics Models* for internal mesh generation by these codes.

2.8.3 Advanced Mesh Generation

Advanced Mesh Generation involves the generation of complex 3D meshes for complex geometries for advanced discretization methods in cutting-edge 3D simulations. Methods for the automatic generation of these types of meshes in largely non-existing and such methods are an active research area. As a result, complex 3D meshes are instead generated using a variety of existing software tools in an iterative process that requires profound expertise in the discretization and numerical methods involved. No general user can generate meshes of this nature and likely will not be able to for many years to come.

Since the generation of complex 3D meshes requires significant expertise and tends to be extremely problem dependent, advanced mesh generation tools will not be developed within VRI. Users of VERA for high-end 3D simulations will be responsible for creating their own meshes outside of VERA. However, VERA will support a limited number of common, 3D mesh data-structure formats.

Examples of the types of meshing tools that VERA users might utilize include CUBIT, Boxer, Spider, Tenasi (from the University of Tennessee at Chattanooga SimCenter), and the mesh generation tools from CD-adapco.

2.9 Materials Interface

The purpose of the *Materials Interface* is to provide the definition for *how* a VERA user or developer may use material properties in a simulation. A material property as discussed here refers specifically to the mathematical model used to represent an intrinsic property of a material as a function of that material's state. An example of a material property is thermal conductivity. The model used to represent the thermal conductivity could just be a constant or it could be a polynomial with several parameters such as temperature and density or possibly a software library. The material's interface does not define the material properties, it only provides a mechanism for using them. In general, VERA will include a basic database of material properties as needed by each physics model.

The functionality provided by the materials interface includes

- how a material property is obtained for use by a physics model,
- how a user specified material is defined, and
- how a user specified material property is defined.

The first bullet is the most relevant to VERA and least relevant to the user. This capability is required by the developers of all the physics models. The second and third bullets are capabilities that have more relevance to the user.

An example use case for the third bullet would be when a user wants to do a fuel performance calculation with a UO2 fuel that has been manufactured in a new way. The user knows that the different manufacturing technique results in different thermal expansion of the pellet, the user would then define a different thermal expansion property not in the database of materials that comes with VERA. The functionality of the second bullet is different from the third bullet if one considers a user who wants to use a brand new material not in the VERA material database, rather than just replacing a property of an existing material. The user would need to define a new material and all its properties insofar as they are required by the physics models.

The design, development, and maintenance of the materials interface is within the scope of VRI. Different physics models will have different requirements for the material properties that need to be used. It will be the responsibility of all of CASL to accurately describe these technical requirements to VRI. The development of the underlying materials properties is the scope of other CASL focus areas and VERA users.

Note that it may be very expensive and therefore unrealistic to require that every physics code that is integrated into VERA take parameters from a centralized materials interface. Therefore, VERA will not necessarily require that all physics codes read from the materials interface. The Reactor-aware FACADEs could restrict the set of material parameters that can manipulated by users and thereby guarantee consistency.

2.10 Data Mapping

The subdomain of *Data Mapping* refers to methods and software for transferring information between different physics models in a multiphysics simulation. The methods and software in this area can be quite sophisticated and often require complex algorithms for transforming information and mapping quantities between different discretization methods on different meshes. These methods must be aware of the parallel machine, the mesh data-structures, and the particular details of the discretizations of all of the physics involved. In some cases, non-trivial nonlinear PDE-based equations must be formulated and solved in order to define the mapping of field data from one physics simulation to another.

In the most general case, defining the data-mapping from one physics model to another requires the full ability to solve a single-physics PDE. In more limited cases, basic mesh interpolations (i.e. not requiring knowledge of the specific discretization methods) between different meshes might suffice. Even in the latter case, remapping of massive amounts of data between different parallel processes might be required which requires sophisticated algorithms and complex software.

With respect to VERA, support for some of the central technology for mesh data-structures and mesh interpolation methods (including parallel data remapping) will be supported. More complex discretization-based data mapping methods will be treated as new type of physics model in VERA.

Data mapping tools cover a broad spectrum and the CASL project will review and leverage a number of external software efforts. The NEAMS project is using the MOAB/MBCoupler software. The Rocstar code has a package called SurfDiver for mapping data between two non-matching meshes on a surface. The Overture code developed at LLNL is an overlapping grid code that requires accurate data transfers. The SIERRA code developed at Sandia contains data mapping algorithms specific to the SIERRA mesh database. Parts of this functionality may be moved into Trilinos/Percept for use in CASL. The Trilinos/Moertel package contains algorithms for mesh tying and contact using Lagrange multiplier constraints. ITAPs is a SciDAC-funded center (FASTMath in SciDAC3) which is developing tools to deliver interoperable and interchangeable mesh, geometry, and field manipulation services that are of direct use to scientific applications. Additionally, the Computational Geometry Algorithms Library (CGAL) and STAR-CCM+ contain algorithms and software of interest.

2.11 Simulation Post-Processing

Simulation Post-Processing involves the extraction of different types of integrated or other reductions of simulation data. For example, the total flux of some quantity (mass, energy, material, etc.) across some boundary might be computed as a post-processed calculation. This type of calculation generally requires detailed knowledge of the discretization and specific physics and would be considered to be an integrated part of software that implements the physics model. However, simpler types of post-processing might include just simple operations on field data such as sums, mins, maxes, averages that requires very little knowledge of the details of the physics or discretization method. These post-processed quantities will be refereed to as *response functions* and these functions are used as the *quantities of interest* (another name for response functions) in VUQ, optimization, and related analysis methods.

Some of the computations for CASL involving response functions might require a substantial physics modeling and solution capability, beyond the existing physics models/solvers. For example, a critical Boron calculation would require its own significant physics modeling capability. Such complex response models would be handled as new physics models integrated into the multiphysics coupling framework.

In general, post-processing would be the domain of the physics model codes themselves and would not be handled by the VERA infrastructure. However, the interfaces for exposing response functions (and adjustable

parameters) will be defined so as to allow VUQ methods (such as are defined in DAKOTA) to drive various analysis studies involving multiphysics VERA simulations. In addition, official VERA support software for some types of discretization-unaware methods could be collected as well (e.g. some tools already exist in Trilinos/STK/Percept and other Trilinos software).

finally, a simpler type of post-processing software involves tools to extract response functions by parsing output files from simulation codes (like a fancy grep). Many such codes exist and one of these could be incorporated into the VERA software stack and supported by VERA developers.

2.12 Multiphysics Coupling

The *Multiphysics Coupling* subdomain refers to general software that can be used to support a variety of methods for coupling different physics models into a multiphysics simulation code. The basic theory associated with the multiphysics coupling algorithms applied in this subdomain is described in [9]. *Multiphysics Coupling* software provides for representing different types of schemes for solving coupled problems, but the linear, nonlinear and transient solver algorithms are implemented in external software such as in Trilinos [1] (e.g. NOX [2], Rythmos, etc.).

Examples of software that may contribute to VERA *Multiphysics Coupling* software include the current LIME [10], Thyra/Rythmos, and AMP; The actual solution may include a combination of more than one of these. Because this software lies at the center of VERA simulators and therefore the fondation of *VERA Base* (see Section 3), we expect it to be licensed as open source software.

All of the multiphysics coupling schemes supported will be represented as functions taking in parameters and returning response functions so that they can be driven by VUQ algorithms such as implemented in DAKOTA. The software will support basic checkpointing and restarting at a fundamental level and should be extensible to any type of multiphysics coupling scheme that could be devised for current or future coupled CASL VERA simulations. The design of this framework will define the basic building blocks for constructing all VERA simulators (and therefore this is a key subdomain of VERA).

This subdomain will consist of entirely Abstract Numerical Algorithms (ANAs) (see Section 2.13). Therefore, this software knows nothing of parallel machines, meshes, or PDE discretizations. Data mapping (see Section 2.10) between different physics models is abstracted behind general functions that allow for any and all arbitrary mechanisms of implementation. Be having is software expressed as ANAs, we avoid having to directly depends on complex software that knows about parallel computing, parallel data distribution, or large-scale data handling or data movement. This allows the multiphysics coupling software to be much more flexible and to be applicable in many different contexts.

2.13 Basic Numerical Algorithms

The subdomain of *Basic Numerical Algorithms* is broken into two other subdomains: abstract numerical algorithms and concrete numerical algorithms.

Abstract Numerical Algorithms (ANAs) were mentioned in Section 2.12 and are comprised of algorithms that only deal with abstract operators, vector spaces, functions on those spaces and algorithms built up from those abstractions [4,3]. This software knows nothing of parallel computing, meshes, or PDE discretizations. Examples of this type of software from Trilinos include NOX, Rythmos, and MOOCHO. All of these Trilinos packages accept problems expressed in the Trilinos standard ANA API called Thyra [3]. Any algorithms expressed in pure Thyra objects almost automatically leads to ANA software.

In addition, gven the black-box nature of most DAKOTA algorithms, DAKOTA would also be categorized as a type of ANA software since it does not see the underlying mesh, discretizations, or details of the parallelization of the physics solves.

Alternatively, *Concrete Numerical Algorithm* software represents support software and implementations of parallel-aware (and possibly mesh and discretization aware) data-structures and algorithms. Example software includes the Trilinos packages Epetra, Tpetra, Ifpack, and ML. This type of software will be heavily leveraged in implementing the guts of physics models and data mapping.

While VERA will strongly leverage this type of software and CASL likely help develop and extend such software, basic numerical algorithm software will not be considered to be a subdomain of VERA.

2.14 Verification and Uncertainty Quantification (VUQ)

The last subdomain of VERA described here is the the VUQ, or *Verification and Uncertainty Quantification* subdomain. From the VERA perspective, *VUQ* is not really a separate software subdomain. Instead, its functionality deals with the support software and algorithms that are used in various verification and UQ studies needed to build a credible foundation for the use of simulation software. The various verification & validation (V&V) processes and the workflows for UQ studies act as requirements for VERA.

The tools that need to be integrated into VERA to carry out V&V, UQ, and related studies impact several other subdomains.

VUQ tools for code verification can require intimate knowledge of the PDE discretization. For example, the method of manufactured solutions (MMS) requires detailed knowledge of the PDE equations and provides complex source terms for verification simulations. ools for MMS would be intimately tied with the specific physics models and would be difficult to reuse across different physics codes.

Various methods for solution verification (such as mesh convergence studies) need to be able to access the massively parallel simulation solution variables on different meshes and compare them. These methods require functionality very similar to the *Data Mapping* subdomain and therefore would be addressed there.

Finally, the higher-level DAKOTA algorithms for driving black-box UQ, sensitivity, parameter, and optimization studies would be classified as *Abstract Numerical Algorithms* as described in Section 2.13. These are shown separately in Figure 1 to highlight their role in driving higher-level studies. The simulation workflow control scripting layer may provide intrisic support for such VUQ methods and workflows.

3 VERA Base

Software comprising *VERA Base* is a subset of the full VERA software stack and will be maximally portable, open source, and hence freely-distributable. These are the components that could be used by anyone to create a PDE-based multiphysics simulation code. For example, external developers could use *VERA Base* to create VERA-compatible components that could then be integrated with native VERA components to form new simulation capabilities.

VERA Base will include at a minimum the application-agnostic aspects of VERA (i.e. functionality that is independent of the nuclear reactor oriented mission of CASL) such as

- "multiphysics coupling framework",
- "data mapping" support,
- "VUQ",
- "simulation post-processing", and
- "simulation workflow control scripting".

In addition, some aspects perhaps more specific to reactor core simulation could be freely-distributed, such as:

- "common reactor input specification",
- "geometry specification",

and potentially even some example physics model implementations.

This set of VERA Base software would be distributed with any release of VERA. For example, there could be a VERA Commercial Open release of VERA containing commercial physics code like STAR-CCM++ coupled with other such codes. Such a release would be open to everyone but would require that users purchase commercial software licenses. As another example, there might be a VERA Proprietary Westinghouse/EPRI release that contains proprietary and export controlled codes. In this case only organizations with the proper permissions would be given these releases. In all of these cases *VERA Base* software would be present providing the key functionality described in the various subdomains it contains.

In order for VERA Base to be used in these various releases, as well as freely distributed releases, it would have to have a very free open-source license, such as BSD. (Note that GPL licences, even LGPL, are considered incompatible with many potential comersial uses and would therefore be inappropriate for inclusion in VERA Base.)

4 Other Architectures Considered

Several other architectures for VERA were considered before arriving at the architecture described above. Here we discuss some of these alternative architectures and why we decided against them.

There is a desire for VERA to be a plug-able infrastructure for nuclear reactor core simulations where the implementations of different component physics models could be plugged in and out using well defined reactor-aware interfaces. Such a design would involve lower-level C/Fortran/C++ interfaces exposing the inputs and outputs from the each of the models and any action behaviors needed for different coupling approaches. These plug-able hookups are often called *ports*. For example, there would be a neutronics port, a thermal hydraulics port, a fuel performance port, as well as other ports for different types of physics and support capabilities. The advantages of an approach like this is that, in theory, someone could come in and unplug an existing physics module and simply plug in a new physics model that takes the same inputs and outputs. Then, all of the data mapping, and other considerations would be handled automatically. The problem with such an approach is that it would be very hard to stabilize the physics port's interfaces and issues like re-meshing would massively complicate the interface and implementations. The more flexible general multiphysics coupling framework based architecture described above allows for a more agile and organic approach for coupling codes but perhaps provides less physics-specific hooks for bringing in new physics implementations.

discussed, place them here if we decide against major parts of them.

5 Conclusions and Future Work

The VERA Architecture description provided in this document gives just the most basic classifications and terminology for VERA and its current scope. Details about each VERA subdomain will be described in other documents (and already are in the cases of the multiphysics coupling framework [9] and the neutronics physics models [6]). All of the components comprising this architecture will be developed at different schedules depending on the (changing) priorities of CASL and CASL customer needs. Also, depending on the needs of CASL challenge problems or real VERA uses, some of these subdomains may never be significantly developed. For example, if it becomes clear that developing a single uniform materials interface is too expensive or just not as critical as was originally thought, it might never be developed. Or, a common reactor input specification might only be developed and used for just the basic core reactor simulator.

Going forward, this VERA Architecture Design will likely be used and modified as follows:

- The specific terminology in this VERA Architecture model will be used in specific AMA requirements documents. This will provide a tie between AMA requirements and specific software efforts in VERA development.
- The set membership and the names of the subdomains may change due to more discussion, thought, and experience. In this case, this VERA Architecture document will be kept up to date with current accepted terminology and definitions.
- As the relationships between the various subdomains are better defined as the software is developed, this knowledge will be reflected in updated diagrams like Figure 1.

In any case, this VERA Architecture document will be kept up to date as the VERA domain model is refined and changed in order to provide an accurate and useful definition of VERA and its terminology (i.e. ubiquitous language).

References

[1] "Sandia National Laboratories." "The Trilinos Project". http://www.trilinos.sandia.gov.

[2] "Sandia National Laboratories." "NOX and LOCA, Object-Oriented Nonlinear Solver and Continuation Packages". http://www.trilinos.sandia.gov/packages/nox.

[3] R. Bartlett, "Thyra linear operators and vectors: Overview of interfaces and support software for the development and interoperability of abstract numerical algorithms.," Tech. Rep. SAND2007-5984, Sandia National Laboratories, 2007.

[4] R. A. Bartlett, B. G. van Bloeman Waanders, and M. A. Heroux, "Vector reduction/transformation operators for linear algebra interfaces to efficiently develop complex abstract numerical algorithms independently of data mapping," 2003. Submitted to *ACM TOMS*.

[5] E. Evans, *Domain-Driven Design*. Addison Wesley, 2004.

[6] T. Evans, G. Davidson, and J. Jarrell, "Design of a neutronics package for multi-physics reactor calculations," Technical Note VRI-11-001, CASL, 2011.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Addison Wesley, 1995.

[8] C. Larman, *Applying UML and patterns*. Prentic Hall PTR, 2nd ed., 2001.

[9] 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.

[10] R. Schmidt, R. A. Bartlett, K. Belcourt, R. Hooper, and R. Pawlowski, "An introduction to LIMEi and its use in coupling codes for multiphysics simulations," Sandia Technical Report In preparation, Sandia National Laboratories.

Distribution

Doug Kothe <kothe@ornl.gov>

CAUTION

This document has not been given final patent clearance and is for internal use only. If this document is to be given public release, it must be cleared through the site Technical Information Office, which will see that the proper patent and technical information reviews are completed in accordance with the policies of Oak Ridge National Laboratory and UT-Battelle, LLC.