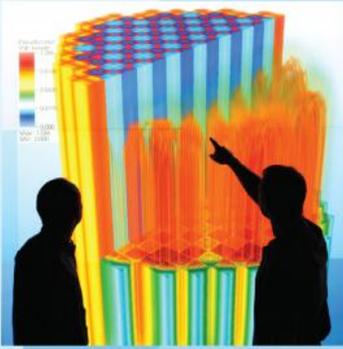




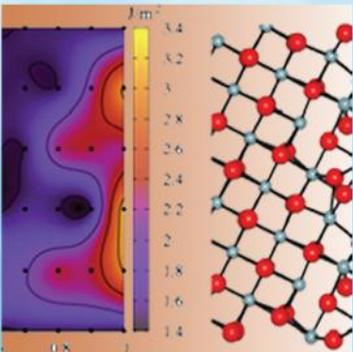
Power uprates  
and plant life extension



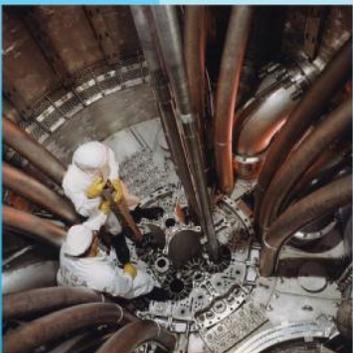
Engineering design  
and analysis



Science-enabling  
high performance  
computing



Fundamental science



Plant operational data

L2:VRI.P2.02

John Turner

ORNL

Completed: 3/31/11



U.S. DEPARTMENT OF  
**ENERGY**

**Nuclear Energy**

# CASL VERA SQA Plan

---

## 1 Basic Information

Project Name	Virtual Environment for Reactor Analysis (VERA)
SRS Registry ID(s)	001578
Software Owner	John Turner
Brief Description of Software	VERA is a code system for scalable simulation of nuclear reactor core behavior. It is a toolkit of components, not a single executable, and contains both legacy and new capabilities encoding a diversity of models, approximations, and algorithms. The software is designed to be scalable from high-end workstations to existing and future HPC platforms.

## 2 Approvals

---

CASL Director

**John A. Turner (jtd)**

Digitally signed by John A. Turner (jtd)  
DN: c=US, o=U.S. Government, ou=Department of Energy,  
ou=Oak Ridge National Laboratory, cn=John A. Turner (jtd)  
Date: 2011.03.31 16:37:17 -04'00'

Date

---

VRI Lead

Date

---

CASL Quality Manager

Date

### 3 Grading Level

Please reference the SBMS subject area "SQA" for definitions of the grading levels.

Safety Software	X
Level A	
Level B	X
Level C	
Research Software	
General Software	
Severe Impact	
Significant Impact	
Moderate Impact	
Minor Impact	

#### 3.1 Software Type

Please reference the SBMS subject area "SQA", exhibit "Software Type Table." If needed, consult with the SAB chair or quality representative for help.

Custom Developed	X
Configurable	
Acquired	
Utility Calculation	

### 4 Description of Software Work Activities

For each of the following software work activities, indicate the level of applicability per SBMS procedure and describe the software development activities that are performed. Please reference the SBMS Exhibit "Safety and General Software Work Activities" for a description of Full vs. Graded requirements.

<b>4.1 Project Management and Quality Planning</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner is responsible for completing a Software Project Management Plan (SPMP) to develop and/or implement an individual software application. The SPMP supports the software life cycle stages and deliverables where appropriate. The SPMP, at a minimum, includes the following:</p> <ul style="list-style-type: none"> <li>• Tasks to be completed;</li> <li>• Person responsible for each task, including responsibility for approvals;</li> <li>• Schedule of milestones and deliverables;</li> <li>• Procurement of services and selection of suppliers;</li> <li>• Costs information;</li> <li>• Schedule for completion of development of software-specific training documentation and end-user training;</li> <li>• Applicable software project risks documented in the SPMP or in other documentation; and</li> <li>• Where more than one organization (including external organizations) is involved in the execution of activities, the responsibilities, interfaces and authority of each organization must be clearly defined and documented</li> </ul>		

Section 5.5 of the CASL proposal to DOE describes the software engineering and quality plans for the project. The software engineering plans are further described in Section 2.2.2 of the proposal.

The proposed quality plan included the following commitments:

- 1) CASL will operate in accordance with the Quality Management System within ORNL's Standards-Based Management System (SBMS) that implements requirements of DOE Order 414.1C, "Quality Assurance" (Ref.2);
- 2) CASL will develop a Quality Plan and submit it to DOE/NE for certification;
- 3) The CASL Quality Plan will be managed by a trained quality control manager who will commit 50% of his/her time.
- 4) Software products developed by CASL partners will also adhere to institution-specific requirements and regulations such as LANL's Software Quality Management policy.

This document is intended to demonstrate compliance to item (1) above. A separate Quality Plan for the CASL consortium will address items 2-4.

The quality assurance (QA) requirements for the U.S. nuclear industry are defined in 10 CFR 50 Appendix B (Ref.3). Each organization that provides services to the industry has a QA program that is reviewed and approved by the U.S. Nuclear Regulatory Commission (NRC). For example, the NRC-approved QA program in Westinghouse is the Quality Management System (QMS) (Ref.5). Other industry partners such as EPRI and TVA have similar QA programs.

The SPMP for VERA is documented in the CASL proposal and Plan of Record (PoR), both available on the [CASL SharePoint site](#). The CASL PoR is reviewed and updated every 6 months. Tasks to be completed are identified as Level 1, 2, or 3 (L1, L2, L3) milestones. The milestones, scheduling, and hierarchy relationships are documented both in the PoR document and in the [CASL Trac site](#).

VRI and VERA milestones and work breakdown tasks are managed via the [VRI Trac Kanban site](#) as tickets. Tickets are assigned a responsible owner and schedule for completion. The processes for working with VRI tickets, including ticket status and review processes, are documented on the Kanban site [here](#).

Costs information is developed and tracked by the CASL Finance Officer and the VRI Team Lead, who are responsible for delivering regular reports and ensuring performance is within budgetary constraints.

Software-specific training documentation deliverables are integrated with the overall development Kanban, and are assigned due dates consistent with milestone delivery.

Project risks and mitigations are documented in the PoR. For a detailed discussion of VERA development risks, see section 4.2 below.

VRI operating procedures, policies (including SQA policies), and guidance are documented on the [VRI Kanban Wiki](#).

The interfaces with external organizations, including roles & responsibilities, are recorded in the work packages, stored on the SharePoint site. Details of how the collaboration intends to address specific work items is embedded in the milestones and in the PoR document. The VRI Kanban site is designed to facilitate distributed software development, integrating contributions from many member organizations in a seamless manner.

<b>4.2 Project Risk Management</b>	Level 1. Full	
	Level 2. Graded	X
	Level 3. No action required	
<p>The Software Owner is responsible for the identification of risks associated with the successful completion of the software project (i.e., development or procurement, and implementation). The software project risks to be considered are those that can cause an adverse impact to the successful completion of the software project, including risks associated with costs, resource availability, schedule, and technical aspects. These risks are documented with mitigation actions and monitored throughout the project by the Software Owner. Risk factors will be managed and tracked as they occur.</p> <p>In the Graded approach, only significant adverse impacts to the completion of the software project are considered.</p>		

Risks to the successful completion of VERA development are identified as part of the biannual planning process, and are documented in the CASL PoR. Risks to successful completion of individual VRI L2 milestones are documented in the CASL Trac system, along with mitigation plans.

<b>4.3 Configuration Management</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner ensures documentation of methods used to control, uniquely identify, describe, and document the configuration of each version or update of software and its related documentation. This documentation includes criteria for configuration identification, change control, configuration status accounting, and configuration reviews and audits.</p> <p>The Software Owner ensures the establishment and/or implementation of a baseline labeling system that uniquely identifies each configuration item, identifies changes to configuration items by revision, and provides the ability to uniquely identify each configuration. This baseline labeling system is used throughout the life cycle of the software, including the development process and during operation until the software is retired.</p> <p>Configuration Change Control: The software owner ensures the establishment and/or implementation of a configuration change control process to ensure proposed changes to the software are documented, evaluated, and approved for release. All changes to software must be formally documented. The documentation must include:</p> <ul style="list-style-type: none"> <li>(a) a description of the change</li> <li>(b) the rationale for the change</li> <li>(c) the identification of affected software application</li> </ul> <p>The defined configuration change control process is used to determine if the software changes have a potential impact on software categorization, user training, user manual, SBMS procedures, internal operating procedures, or software security as applicable. The changes will be formally evaluated and approved by the organization responsible for the original design, unless an alternate organization has been given the authority to approve the changes. The software owner provides acceptance testing and approval for the release of software as outlined in the section on Software Verification and Validation.</p>		

**Configuration Status Accounting:** The Software Owner ensures the establishment and/or implementation of a process for maintaining records and reports related to the configuration of the software. The focus of this process is on reporting information needed to maintain integrity and traceability of a controlled Configuration Item and its associated documentation throughout the life cycle of the software.

Status accounting records and reports must be available that provide the current status and history of configuration items. The following minimum data elements must be tracked and reported for each configuration item, its approved versions, the status of requested changes, and the implementation status of approved changes. The level of detail and specific data required can vary according to the information needs of the project and the customer.

**Configuration Audits:**

**Full Approach:** The Software Owner ensures periodic assessments are conducted to verify that the change request documentation is consistent with the implemented software change.

**Graded Approach:** Optional performance of configuration audits and reviews.

VERA makes use of the Git source code repository system, which is widely used in the software development industry. Git repositories provide a complete history and full revision tracking of all source code files. Each revision to the software is labeled.

The component software used by VERA resides in a variety of source code repositories. An index of these components and their repositories is maintained on the CASL VRI Kanban Wiki here.

Changes to the software are controlled via the VRI Kanban project management process, hosted in Trac. Proposed changes, bugs, and new features are entered into the Trac backlog as tickets and are assigned tracking numbers and owners. Tickets provide fields for the change description, rationale, and other information. Each ticket is marshaled through the development process by assigning a status value, which encodes the workflow steps. Comments, updates, links to data, and documents are associated with each ticket. A complete history of all changes to tickets is tracked in a backing database. Tickets are not assigned a "complete" status until they have been reviewed by the VRI team and approved by the VRI Kanban Lead and the FA Leader.

The VRI Software Development Life Cycle follows a hybrid Scrum + Kanban ("Scrumban") approach. Software releases occur at the end of 6 month development "sprints". Prior to each release, a review session is held to review the completed tickets and authorize their delivery to the AMA FA for acceptance testing, to examine the processes, procedures and policies used during the previous sprint for modifications, and to plan for the next sprint.

AMA performs an independent suite of tests on each release to determine if requirements are satisfactorily met. In the event of defects being detected, they are reported back to the VRI team and are entered into the Trac backlog as tickets.

Documentation, development notes, policies and procedures are configuration controlled via the CASL VRI Wiki and CASL SharePoint document repositories, as described in the CASL Records Management Plan.

Software development processes, procedures, and policies are reviewed every month for continuing suitability and improvement during a Retrospective meeting. Changes to these processes are recorded

Template version 2.0, for General Software

in the Wiki. The Wiki structure always presents the most recent version of a document; obsolete versions are automatically removed from service.

Changes to VERA are associated with documented acceptance tests validating and verifying the change.

<b>4.4 Procurement &amp; Supplier Management</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner is responsible for the management of the process to ensure that the supplier for commercial off-the-shelf software and other types of acquired software and services (e.g., software as a Service, Cloud) are capable of providing software that satisfies the technical and quality requirements identified in the procurement documentation.</p> <p>The Software Owner ensures the development of procurement documentation that includes the technical and quality requirements for the software. Items to be included (as applicable) are:</p> <ul style="list-style-type: none"> <li>Specifications for the software features, including requirements for safety, security, functions, performance, standards and target environment (e.g., hardware, operating system).</li> <li>Process steps used in developing and validating the software, including any documentation to be delivered.</li> <li>Requirements for the supplier to provide formal on-site, off-site or on-line training for users, if requested.</li> <li>Requirements for the supplier to provide a process to (a) notify users of defects, new releases, or other issues that impact the operation of the software and (b) allow users to report defects, problem resolution and request assistance in operating the software.</li> <li>Requirements for labeling new releases or upgrades of software to uniquely identify changes by revision.</li> </ul> <p>The Software Owner is responsible for ensuring the quality of the software that is being purchased and that it meets or exceeds the procurement requirements. This includes a review of the supplier's quality assurance program through supplier assessment, supplier self-declaration, third-party certification (e.g., the International Organization for Standardization, Underwriters Laboratories, and Software Engineering Institute), acceptance based on key characteristics (e.g., large user base), or other similar methods.</p> <p>Note: The Software Owner is responsible for ensuring the delivered product has been validated and meets the intended requirements prior to releasing into production environment. (Refer to section on Software Verification and Validation).</p>		

Procurement for VRI uses the ORNL procurement process as described in [SBMS](#). A full-time CASL contract coordinator handles procurement and management of contracts with partner institutions, board and council members, and other individuals to ensure that institutional guidelines are identified and followed.

At present, software components for VERA have been and are being obtained from a variety of sources based upon the capability, reputation, and availability of the source code. In some cases, the acquired code is delivered with unit tests and/or other test cases that are migrated to the VERA/VRI test base. In the case of older, legacy code, the programs chosen for inclusion in VERA have been tested through many years of application in research and industry. This latter class of programs presents a special challenge to VRI, and research is underway to determine effective methods of encapsulating and refactoring legacy code to introduce modern software engineering test practices.

In all cases, source code acquired by VRI for inclusion in VERA is placed under the same change control process as in-house code.

As a future improvement, a process will be instituted to properly vet acquired software against requirements and QA provenance, and develop migration strategies that introduce CASL SQA standards.

Template version 2.0, for General Software

<b>4.5 Requirements Identification &amp; Management</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner ensures the identification of software requirements and documents those requirements in a software requirements specification, procurement contracts and/or acquired software agreements. These requirements include functional requirements; performance requirements; security requirements, including user access control; interface and safety requirements; installation considerations; and design constraints, where appropriate. These requirements are documented in a software requirements specification, procurement contracts and/or acquired software agreements.</p> <p>Once the software requirements have been defined and documented, they are maintained to ensure the correctness of the software and used for verification and validation activities. Software requirements must be traceable throughout the software life cycle.</p>		

Requirements for VERA are formally developed and documented by the Advanced Modeling Applications (AMA) Focus Area. AMA is the customer for VERA until such time as releases of the code base are made available to third parties who are not part of the CASL consortium. The documented requirements are handed off to the VRI, who produces an implementation plan. After negotiation and agreement on features to be delivered and their priority, requirements are broken down into work items and entered into the VRI Kanban TRAC backlog.

Requirements are implemented using a hybrid Kanban/Scrum project management technique, in which features, bugs, and supporting work items are entered into a backlog of work tickets and implemented on the basis of priority.

<b>4.6 Design &amp; Implementation: Design Description</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner is responsible for ensuring that the design of the software is documented. The software design elements identify the operating system, function, interfaces, performance requirements, installation considerations, design inputs, and design constraints where appropriate. Reviews of the design and code must be performed. Formal developer testing (e.g., unit, integration, etc.) that includes functional, structural, timing, stress, security, and human-factors testing are planned and performed where appropriate, and the results are documented.</p> <p>Once the software design description (including interfaces and data structures) has been defined and documented, it must be maintained to ensure the correctness of the software and will be used for verification and validation activities. The software design description is documented in a software design document or included within the software requirements document.</p> <p>Note: The design activities and documentation should be adequate to fully describe how the software will interface with other system components and how the software will function internally. Data structure requirements and layouts may be necessary to fully understand the internal operations of the software.</p>		

The VRI team will develop a high-level design document that describes the overall system architecture and roadmap for VERA, due by July 2011 (Reference VRI Kanban ticket [2140](#)). This design will be change controlled, and will be reviewed monthly during the VRI Review, Retrospective, and Planning process. However, the primary design artifacts in a Lean/Agile software development process are captured in the

Template version 2.0, for General Software

software itself and the natural byproducts of software development processes. Requirements are primarily stated as L1, L2, and L3 milestones in the CASL Milestones Trac project and CASL VRI Kanban tickets are linked to these milestones. With the integration of Trac and Git, full linkage from CASL Milestones, through VRI Kanban tickets, down to individual Git commits is provided. Requirements are codified as version-controlled automated unit, acceptance, and verification tests. Changes in requirements are reflected by changes in the automated test suite (e.g., new requirements result in new tests, no longer valid requirements result in tests being removed). The design of the software is embedded in the domain model which is codified in the software sources. Detailed design discussions are captured on Trac wiki pages and inside of Trac tickets in the CASL VRI Kanban Trac Site. Detailed design is documented in the sources itself and is extracted using tools like Doxygen. The understandability and maintainability of the software (including algorithms and data structures) is primarily provided by keeping a clean code design and structuring using Agile Emergent Design using Structured Refactoring using good software design and coding techniques and all protected with high quality unit, integration, and verification tests. Low-level supplemental documentation will only be provided if good software design and coding practices are not adequate. The approaches described above are the only realistic ways that design can be done and documented on a project like CASL.

The design of models and theoretical constructs to be implemented in VERA are documented by the MNM Focus Area, and are associated with implementation work tickets using the wiki.

<b>4.7 Design &amp; Implementation: Design and Code Review</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
The Software Owner ensures that reviews of the design and code are performed.		

Monthly Review, Retrospective, and Planning (RRP) sessions review every aspect of VRI development, and may include design.

The VRI team will investigate possible policies on code and design reviews (for changes to legacy code and new code development), and implement a strategy that is compatible with modern Lean/Agile best practices.

<b>4.8 Design &amp; Implementation: Developer Testing (Unit, Integration)</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
Formal developer testing (e.g., unit, integration, etc.) that includes functional, structural, timing, stress, security, and human-factors testing are planned and performed where appropriate, and the results are documented.		

CASL VRI software quality assurance policies are posted in the VRI Kanban wiki [here](#). These policies, by default, mirror those in use by the Sandia Trilinos project.

Template version 2.0, for General Software

VERA is developed using continuous integration (CI), in which the code base is built many times a day and a suite of automated tests is run on the primary development platform(s) (Linux Intel and GC). Defects detected during the CI builds are immediately communicated to developers through an automated reporting system (implemented with CDash). It is required to test all new or modified Primary Stable code prior to checking into the source repository. Portability testing and other more detailed tests are run in nightly builds.

Some integration testing is performed as part of the daily test suite. Rigorous integration and human factors testing will be performed by the AMA and VUQ focus areas, as part of their V&V plan.

<b>4.9 Failure Analysis</b>	Level 1. Full	
	Level 2. Graded	X
	Level 3. No action required	
<p>The Software Owner is responsible to ensure a software failure analysis is performed and documented.          Note: The techniques used to evaluate potential software failures may include a software or system level failure modes and effects analysis, fault-tree analysis, event-tree analysis, cause-consequence diagrams, interface analysis, nuclear safety cross check analysis, and hazard and operability studies.</p>		

For the near term, VERA will be used in a tightly controlled, research only environment. The potential risks to software failure are relatively minor, and are comprehended as part of the project risk analysis.

<b>4.10 Verification &amp; Validation</b>	Level 1. Full	
	Level 2. Graded	X
	Level 3. No action required	
<p>The Software Owner is responsible to ensure software verification and validation activities are performed and documented. Reviews and inspections of software deliverables requirement specifications, procurement documents, software design documents, code modules, test results, training materials, user documentation, and processes that guide the software development activities will be performed as applicable. Verification of the software design, using inspections, walkthroughs, desk checks, code reviews, or other methods must be completed prior to approving the software for use where appropriate.          Note: This verification may have been performed as part of the Software Design and Implementation work activity.</p> <p>Traceability of the software requirements to the software design is performed where appropriate.</p> <p>Note: Required tests should be controlled under an isolated environment for testing software from the production environment (e.g., separate physical device, virtual environment, offline production system, production system with safeguards, etc.).</p> <p>The Software Owner is responsible for ensuring software acceptance testing for the software is performed and documented. Acceptance testing includes functional testing, performance testing, security testing, stress testing, and load testing where applicable. The testing must be performed prior to approval of the software for use. Acceptance test cases and procedures, including expected results, are documented. Test results are to be documented and approved prior to production release (per the configuration management process). All deliverables are under configuration control.</p>		

V&V activities for VERA are distributed across the focus areas. A detailed V&V plan for VERA has been described elsewhere in the Initial Validation Plan for CASL Project document (AMA.VAL.Y1.02).

Template version 2.0, for General Software

Tractability of requirements to design is captured by linkages between CASL Milestones and CASL VRI Kanban Tickets in the Trac projects.

<b>4.11 Problem Reporting &amp; Corrective Action</b>	Level 1. Full	X
	Level 2. Graded	
	Level 3. No action required	
<p>The Software Owner is responsible to ensure a process is implemented for documenting software problems and corrective actions to address software errors, failures and their resolution. The reporting and corrective action system will include (1) methods for documenting, evaluating, and correcting software problems; (2) an evaluation process for determining if a reported problem is a defect; and (3) the roles and responsibilities for disposition of the problem reports, including notification to the originator of the results of the evaluation.</p> <p>If the noted problem is a defect, the problem reporting and corrective action system must:</p> <ul style="list-style-type: none"> <li>correlate the defect with the appropriate software engineering elements,</li> <li>identify the potential impacts and risks to past, present, and future developmental and operational activities, and</li> <li>support the development of mitigation strategies.</li> </ul> <p>This formal implementation must include documentation and tracking to closure of any problems reported for the software and authorization to perform the corrective action.</p> <p>Note: After a defect has been noted, users should be apprised to ascertain any impacts upon previous decisions.</p>		

Any CASL member or CASL stakeholder can create a new CASL VRI Kanban ticket and report a defect or request a new feature.

Bug reports are entered into the Trac tool as a new ticket, and are prioritized based on the severity of the defect. The status tracking and documentation of the bug are handled using the ticket status, comment, and attachment features.

A set of formal policies will be developed that describe the process for evaluating and responding to bug reports, including impact analysis and reporting to users.

<b>4.12 Training of Developers, Operators &amp; Users</b>	Level 1. Full	
	Level 2. Graded	X
	Level 3. No action required	
<p>The Software Owner is responsible for ensuring periodic reviews of the completion of training, education, and/or qualification requirements for all staff involved in software development, testing, use, and the evaluation of software are performed and documented. This includes a position description, qualification criteria, or a list of training courses along with verification of successfully meeting the knowledge requirements.</p>		

VRI will develop a user manual that will be updated with each formal release. Hands-on user training is also provided on a (semi) regular basis by the CASL consortium. Informal developer training will be done through reading groups and peer-to-peer training from knowledgeable developers.

## **4.13 Security**

Elements of VERA are subject to export control and/or NDA. Source code, input data, and output data are controlled through GForge project membership and password-protected permissions and Unix groups are used to protect such data on various internal ORNL CASL-related machines. Documents which may contain controlled or protected information will be handled per the CASL Records Management Plan. A Technology Protection Plan will be put in place which describes the processes and systems used to manage information security.

VERA as a software application executes in a closed environment, and does not communicate with data or application servers via the open internet. As such it presents no serious security risks.