

**Configuration Management Plan
for the SCALE Code System**

Prepared by
B. T. Rearden, S. M. Bowman, and J. P. Lefebvre

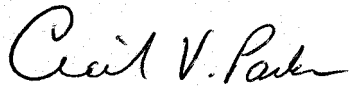
Reactor and Nuclear Systems Division (RNSD)
Oak Ridge National Laboratory

Date Revised: May 22, 2013

Approvals:

SCALE Project Leader

Date



Director, Reactor and Nuclear Systems Division

Date

Quality Manager, Performance Analysis and Quality

Date

1.0 INTRODUCTION

- 1.1 SCALE is a modular computational system for nuclear safety analysis and design consisting of a driver module, control modules, functional modules, utility modules, data libraries, and subroutine libraries. The SCALE system draws heavily from basic radiation transport and nuclear data processing methods technology developed at Oak Ridge National Laboratory (ORNL) since the 1960s and continuing with state-of-the-art methods today. It is designed to provide modeling and simulation capabilities for cross-section processing, criticality safety, reactor physics, radiation shielding, radioactive source term quantification, including radiation spectra, decay heat, nuclide generation and decay, as well as sensitivity and uncertainty analysis. The overall goal of the SCALE project has been to develop easy-to-use analytical tools that are automated to perform data processing and analysis using well-established computer codes and data libraries that are developed and maintained within a quality-assurance framework.
- 1.2 Legacy SCALE functionality performs serial calculations and consists of control modules, functional modules, utility modules, a driver module, shared subroutine libraries, data libraries, a runtime environment, and graphical user interfaces.

The driver module remains active at all times and transfers the control and functional modules to and from the central processor unit. A control module drives each analytic sequence. The control module automates the necessary data processing, generates the input to the functional modules, initiates module execution in proper order, and performs any needed post-processing. The functional modules are the analytical codes that provide the specific capabilities required to solve the problem of interest. They may be executed as part of an analytical sequence when called by a control module, or in stand-alone mode.

Utility modules allow users to combine problem-dependent cross-section libraries, convert their non-SCALE cross-section libraries to obtain libraries with the format and nuclide identifiers required by a SCALE module, or perform other editing, data checking, or post-processing functions.

The subroutine libraries contain source code that are accessed by multiple modules in SCALE. Whenever a change is made to one of these subroutines, every module that accesses that subroutine must be re-linked with the modified subroutine. The source code for each of the impacted modules remains the same, but the executable code is changed.

Within a legacy SCALE calculation, extensive file input/output (I/O) is required to enable communication between the numerous functional and control modules. Additionally, widespread use of global variables limits parallel implementation.

- 1.3 Many features introduced after the release of SCALE 6.1 in 2011 are based on modern software architecture to enable multiple functions to be performed from a single modular executable program capable of parallel operation. An object-oriented framework is established to enable modular design, reusable components through extensibility, and portability. Efficiency gains are realized by minimizing file I/O throughout the calculation.
- 1.4 SCALE data libraries are validated data files that contain nuclear cross-section data, material compositions, or physical properties. The data can be easily accessed via keyword input to a control or functional module.
- 1.5 Since SCALE is a modular code system with interrelated modules and libraries developed over several decades with differing technologies and design philosophies, configuration management is a complex and challenging task. An agile continuous integration model is described in this configuration management plan (CMP) to enable developers to implement changes to SCALE in a manner that ensures that features integrated into the system do not prohibit the operation of existing features and are immediately available for deployment to sponsors and end users. Changes to the system (computer code, data, and documentation) will be controlled by this CMP under the direction of the SCALE Project Leader.

2.0 PURPOSE

The purpose of this plan is to describe the methods used for proposing, approving, implementing, testing, and deploying changes to the SCALE computational system, which is controlled by Reactor and Nuclear Systems Division (RNSD).

3.0 SCOPE

This plan applies to the modification of existing software or data or the development of new software or data in the SCALE computational system controlled by RNSD on the designated ORNL computer systems and deployed to end users through the Radiation Safety Information Computational Center (RSICC) and further redistributed through international centers as designated by RSICC.

4.0 REFERENCES

- 4.1 *Quality Assurance Plan for the SCALE Computational System*, QAP-005.
- 4.2 *SCALE Procedure for SCALE Discrepancy Reports*, SCALE-CMP-004.
- 4.3 *SCALE Procedure for Feature Changes*, SCALE-CMP-013

5.0 REQUIREMENTS

Documentation and verification are required for all software and data modified or developed in the SCALE computational system. A validation report may be required for modifications or additions to SCALE, as determined by the SCALE Project Leader.

6.0 DEFINITIONS

- 6.1 **Computer Code** - A set of instructions that can be interpreted and acted upon by a programmable digital computer.
- 6.2 **Source Code** - A computer code in its originally coded form, typically in text file format. For programs written in a compilable programming language, the uncompiled program.
- 6.3 **Object Code** - A computer code in its compiled form. This applies only to programs written in a compilable programming language.
- 6.4 **Executable Code** - The user form of a computer code. For programs written in a compilable programming language, the compiled and loaded program. For programs written in an interpretable programming language, the source code.

- 6.5 **Data Library** - A data file for use with an executable code that is not intended for modification by the user.
- 6.6 **Sample Input** - Input data for a designated sample problem used to verify correct installation and operation of the code system.
- 6.7 **Test Case Input** - Input data for a test case used to verify a modification to a module or a data library.
- 6.8 **Software** - Computer codes and data in electronic file format.
- 6.9 **Design Requirements** - Description of the methodology, assumptions, functional requirements, and technical requirements for a Feature.
- 6.10 **Discrepancy** - The failure of software or data to perform according to its documentation.
- 6.11 **Validation** - Assurance that a model as embodied in a computer code is a correct representation of the process or system for which it is intended. This is usually accomplished by comparing code results to either physical data or a validated code designed to perform the same type of analysis.
- 6.12 **Verification** - Assurance that a computer code correctly performs the operations specified in a numerical model or the options specified in the user input. This is usually accomplished by comparing code results to a hand calculation or an analytical solution or approximation.
- 6.13 **Version Number** - A number of the form X.Y.Z used to identify the version of a computer code. A version number is assigned when the computer code is implemented for production use. For modifications between baselined versions of SCALE, the last field is increased by one. When a new SCALE baseline is established, the X.Y fields are increased to the new version number of the entire SCALE computational system.
- 6.14 **SCALE Development Repository** – This is the working area for SCALE developers to introduce new features or partial features. This area is maintained under a version control system. Any updates to this area are cataloged in an electronic system. Each update triggers a build and execution of a test suite on each designated platform. Any failures of the test suite are reported to the cognizant developer for resolution.
- 6.15 **SCALE Staging Repository** – This is a testing area for developers and reviewers to assess the quality of new features in a stable environment. Features are migrated to this area only after they pass all tests in the Development Repository. Features in

this area must always pass all tests on all designated platforms. Features in this area are subjected to additional testing until they are certified as ready to ship.

- 6.16 **SCALE Production Repository** – This is the pre-release area for new features. Only features that have completed all quality assurance requirements are migrated to the Production Repository.
- 6.17 **Feature** – A user-accessible capability in SCALE that can include updated software and/or data.
- 6.18 **Kanban** –Literally a Japanese word for "signboard" or "billboard", is a concept related to lean and just-in-time (JIT) production. According to its creator, Taiichi Ohno, Kanban is one means through which JIT is achieved. Kanban is a scheduling system that helps determine what to produce, when to produce it, and how much to produce.
- 6.19 **Change Control Board** – A designated group of individuals who make decisions regarding major Feature revisions.

7.0 RESPONSIBILITIES

- 7.1 **SCALE Project Leader** - The person responsible for managing the maintenance and development of the SCALE computational system. The SCALE Project Leader is responsible for approval of all modifications to SCALE. As such, the SCALE Project Leader is responsible for the configuration management and quality assurance of the SCALE software system. The RNSD director may sign for the SCALE Project Leader in his absence.
- 7.2 **SCALE Leadership Team** – The SCALE Leadership Team consists of the SCALE Project Leader, line managers, program managers, and developers as designated by the SCALE Project Leader. The Leadership Team meets regularly to discuss the current status and make programmatic and managerial decisions regarding SCALE. For major functional changes to SCALE, the Leadership Team serves as the Change Control Board (CCB).
- 7.3 **Developer** - The staff member responsible for the development, maintenance, verification, validation, and documentation of changes to a module or a data library.
- 7.4 **Software Quality Assurance (SQA) Coordinator** - Staff member responsible for the implementation, protection, and maintenance of the SCALE software system.
- 7.5 **Technical Reviewer** – Staff member responsible for ensuring that functional changes to software or data libraries perform as expected and are adequately documented.

- 7.6 **Code Reviewer** – Staff member responsible for ensuring that software and data libraries are well designed and well implemented.

8.0 PROCEDURE

8.1 Configuration Identification and Integrity

- 8.1.1 Each computer code and data library is identified by a unique name and version. These are displayed in the code output in order that users may verify the code version used to produce the results.
- 8.1.2 The SCALE software and data Configuration Control Lists (CCLs) identify the system baseline and are maintained by the SQA Coordinator for each computer system designated by the SCALE Project Leader. Copies of the CCL's are kept in a file system and updated by the SQA Coordinator whenever a change is made. The CCL's are signed and filed by the SQA Coordinator and SCALE Project Leader on a quarterly basis. They include all production source code, data libraries, and sample input files.

The source code, data libraries, runtime environment, build system, sample input files, and expected output results are maintained in centralized repositories. The software repository provides documentation of the version, revision date, and location of each component. All revisions to the repository are referenced in SCALE Quality Assurance Feature Cases in an electronic tracking system. The SQA Coordinator also maintains a high-level log of all completed and outstanding revisions in the SCALE Change Log (SCL). Discrepancies are reported as Bugs in the electronic tracking system, and are logged by the SQA Coordinator in the SCALE Discrepancy Log (SDL).

- 8.1.3 The production version of source modules, object modules, executable modules, data libraries, runtime environment, build system, sample input files, and expected output results are protected by the SQA Coordinator to ensure their integrity from unauthorized modification. The SQA Coordinator is responsible for appropriate backups of these files.
- 8.1.4 Stand-alone computer codes supplemental to the SCALE system but not normally included therein may be included and controlled by this procedure if so designated by the SCALE Project Leader.

8.2 Configuration Change Control

- 8.2.1 All changes to SCALE software or data must be planned and implemented through an approval and review process using a graded approach. An

electronic system that implements the Kanban process is used to track the progress of each Feature through design, implementation, documentation, testing, and deployment. The Kanban steps implemented for change control are:

- 8.2.1.1 Proposed – Requirements for feature, testing, and documentation, collectively called the Feature, are determined and reviewed.
- 8.2.1.2 Approved – SCALE Project Leader acknowledges need for the Feature, adequacy of proposal, and availability of resources for implementation.
- 8.2.1.3 In Progress – Developers implement the Feature in the SCALE Development repository only after the proposal is approved.
- 8.2.1.4 In Testing – Technical Reviewers and/or Code Reviewers assess the Feature in the SCALE Staging repository.
- 8.2.1.5 Ready to Ship – After approval by the SCALE Project Leader, the SQA Coordinator migrates the Feature to the SCALE Production repository.
- 8.2.1.6 Deployed – At a time determined by the SCALE Project Leader, all Ready to Ship Features are deployed in a beta release or a general release of SCALE.

8.2.2 Changes to SCALE are conducted according to SCALE-CMP-013, *SCALE Procedure for Feature Change*. Only the SCALE Project Leader may change the Kanban status of a Feature after sufficient review and testing is conducted. An SCL summarizing all changes to the system is maintained by the SQA Coordinator.

8.3 **Discrepancy Reports and Corrections**

- 8.3.1 Problems encountered with SCALE are reported to the SCALE Project Leader, recorded in an electronic tracking system, and tracked to resolution.
- 8.3.2 Issues identified as discrepancies are resolved according to SCALE-CMP-004, *SCALE Procedure for Discrepancy Reports*.
- 8.3.3 The SCALE Project Leader ensures that RSICC, users, and sponsors are notified of any corrective actions and their impact on users via e-mail notices, web postings, and/or the SCALE Newsletter as deemed appropriate.
- 8.3.4 An SDL is maintained by the SQA Coordinator.

8.4 **Configuration Assessments**

- 8.4.1 A configuration assessment of the SCALE code system is performed at least

once each year by a cognizant staff member designated by the SCALE Project Leader. The purpose of the assessment is to verify compliance with this configuration management plan.

8.4.2 The results of the assessment are reported in writing to the SCALE Project Leader.

8.4.3 The SCALE Project Leader ensures that any necessary corrective actions resulting from the assessment are completed and documented in accordance with the SCALE QA Manual.

8.5 Status Accounting

8.5.1 The SQA Coordinator maintains a docket of all completed and outstanding Features and Bugs in the SCL and SDL logs, respectively.

8.5.2 The SQA Coordinator provides the SCALE Project Leader with a monthly report of all docketed items that identifies items that were resolved.

8.5.3 The SQA Coordinator compiles and transmits a report to RSICC on a quarterly basis to account for all docketed items that were resolved during that time period.