Preparation for a PROTEUS Software Quality Assurance Plan

Nuclear Science and Engineering Division

About Argonne National Laboratory

Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's SciTech Connect (<http://www.osti.gov/scitech/>)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd Alexandria, VA 22312 www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: orders@ntis.gov

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

Preparation for a PROTEUS Software Quality Assurance Plan

prepared by
Emily R. Shemon
Nuclear Science and Engineering Division, Argonne National Laboratory

September 30, 2018

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC

ABSTRACT

PROTEUS is the neutron transport solver package developed at Argonne National Laboratory under the DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program. For the purposes of this report, we consider the PROTEUS package to include the three solvers PROTEUS-NODAL, PROTEUS-MOC, and PROTEUS-SN, as well as the PROTEUS mesh tools and utilities.

As the developer base for PROTEUS has expanded to include university and other national lab contributions and the user base includes industry and federal programs outside of DOE-NEAMS, a more rigorous procedure for assuring the software quality of PROTEUS is necessary. This report, which can be considered the predecessor to a formal software quality assurance (SQA) plan, lays out the basic framework needed to support implementation of a formal SQA plan. In particular, the relocation of all codes to a single version control system, implementation of rigorous unit and regression tests, improved configuration management system, and adoption of a database system to track bugs, feature requests, and development work are all required in a framework that facilitates software quality. This report therefore presents initial milestones to be completed in order to support implementation of a rigorous SQA plan in the near future.

TABLE OF CONTENTS

Abstract	i
Table of Contents	ii
1. Introduction	1
2. PROTEUS Overview	1
3. Baseline Requirements.....	2
4. Preparing PROTEUS for SQA.....	3
4.1 Version Control.....	3
4.2 Unit and Regression Tests.....	4
4.3 Configuration Management	5
4.4 Automated Testing	5
4.5 Software Development Life Cycle.....	5
4.6 Availability of Qualified Personnel and Resources	6
5. Preliminary Software Quality Assurance Plan.....	7
6. Next Steps	8
References	9

1. Introduction

PROTEUS [1] is the neutron transport solver package developed at Argonne National Laboratory (Argonne) under the DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program. For the purposes of this report, we consider the PROTEUS package to include the following individual modules / codes:

- PROTEUS-NODAL solver [2]
- PROTEUS-MOC solver [3]
- PROTEUS-SN solver [4]
- PROTEUS Mesh Tools [5]
- PROTEUS Utilities
- Cross Section API [6]

As the developer base for PROTEUS has expanded to include university and other national lab contributions and the user base includes industry and federal programs outside of DOE-NEAMS, a more rigorous procedure for assuring the software quality of PROTEUS is necessary. This report, which can be considered a predecessor to a formal software quality assurance (SQA) plan, lays out the basic framework needed to support implementation of a formal SQA plan. In particular, the relocation of all codes to a single version control system, implementation of rigorous unit and regression tests, improved configuration management system, and adoption of a database system to track bugs, feature requests, and development work are all required in a framework that facilitates software quality. This report therefore presents initial milestones to be completed in order support implementation of a rigorous SQA plan in the near future. [Note that SQA procedures for the MC²-3 code [7] are not covered in this document. MC²-3 will be addressed as part of a formal SQA program in FY19.]

2. PROTEUS Overview

PROTEUS is a complete set of tools for performing reactor physics analysis of various reactor types. A simple viewgraph of PROTEUS is included as Figure 1. Three solvers, PROTEUS-NODAL, PROTEUS-MOC, and PROTEUS-SN are available. The PROTEUS-NODAL code is a modern day replacement for DIF3D-VARIANT [8] and uses homogenized assembly representation for quick runtimes. The PROTEUS-MOC code is a three-dimensional solver based on the method of characteristics in the 2D plane, and the discontinuous Galerkin finite element to couple the 2D planes axially. The PROTEUS-SN code is a three dimensional solver based on the continuous Galerkin finite element method (space) and discrete ordinates (angle) discretizations of the even parity transport equation.

The two main inputs to these solvers are meshes and multigroup cross sections. PROTEUS Mesh Tools provide a simple, clean interface (no external package reliance) to create finite element meshes for PROTEUS-MOC and PROTEUS-SN. The PROTEUS-NODAL geometry is based upon a semi-structured grid and thus a simplistic geometric input capability is included specifically for it. The MC²-3 multigroup cross section generation code and the cross section API (CSAPI) are available to transform material interaction cross sections into discretized multigroup form for use in the deterministic solvers. MC²-3 can be run as a pre-processing step, or the CSAPI can be run during the transport solver simulation to calculate multigroup cross sections on-the-fly.

After analysis is finished, a variety of utility tools are available to post-process the data. For example, tools are available to convert text output to visualization (HDF5 or VTK) formats, and to print tables of values for different regions. More utility tools are likely to be added in the future to enable the user to retrieve data easily.

Figure 1. High-Level View of PROTEUS.

The different solvers in PROTEUS are coupled, or in the process of being coupled, to other frameworks and physics solvers. At this point in time, we consider only the codes in the above viewgraph for implementation and testing of our initial software quality assurance procedures as these codes are all developed by Argonne National Laboratory. Eventually, once a SQA program is in place, support should be extended to test coupled instances of PROTEUS components.

3. Baseline Requirements

In order to prepare for compliance with a software quality assurance plan, several conditions on the software development life cycle must be met.

1. Source code is tracked using an electronic version control system.

2. A suite of regression and unit tests is available to verify all important functions of the code.
3. Configuration management controls are in place to control and record the build settings as well as identify the version number of the software.
4. Execution of verification tests for different configurations is automatically performed on a regular basis (either scheduled or continuous).
5. The software development life cycle includes specific processes to direct the handling of new features, bug fixes, and releases to users. The processes assign parties to handle each step of the life cycle. An electronic database system (i.e. Manuscript/FogBugz, SVN Trac, or the provided interface in GitLab) is ideally employed in order to reduce paperwork and/or email traffic while keeping a record of all changes to the code base.
6. Personnel and computing resources are available on a consistent basis, and they are committed to complying with the SQA plan.

Once these baseline requirements are in place, the software quality assurance plan can be followed accurately and efficiently for the introduction of new features, bug fixes, and software releases. The following section presents the status of each of these requirements as well as next steps required to be considered complete.

4. Preparing PROTEUS for SQA

In this section, the baseline requirements are discussed as they pertain specifically to PROTEUS.

4.1 Version Control

To properly implement a rigorous software quality assurance plan, the source code must be stored and tracked using an electronic version control system. All PROTEUS source code is currently under version control in different locations including Subversion (SVN) and Git repositories. The existence of multiple repositories was done to enable access for external collaborators as well as meet export control restrictions. However, this model is no longer desirable because it allows common libraries to be located in different places, increases the probability of divergence of source code, and increases the complexity of testing and accessing the different modules.

Moving all source codes to a common repository will simplify maintenance of the code significantly while enabling collaboration and maintaining access controls. The only complexity introduced by moving to a common repository is that the SVN-based SHARP toolkit, which currently hosts PROTEUS-SN, will need to change its access and testing procedures.

The decision of where to permanently store PROTEUS is under discussion with computing authorities in the Computing, Energy and Life Sciences (CELS) Division at Argonne. At the time of this report, there is no approved repository with encryption (needed for export controlled codes) that is also accessible from outside the Argonne network. Due to PROTEUS's many university collaborators, external access off Argonne's network is crucial. As of September 2018, the Argonne CELS representative communicated that the XGitLab repository hosted by Argonne at <https://xgitlab.cels.anl.gov> is an acceptable intermediate location to host the PROTEUS source code and verification tests while the appropriate repository is being developed, provided access controls are implemented per export control requirements.

Table I. PROTEUS components and repository locations.

Component	Current Repository	Intermediate	Future
PROTEUS Basic Libraries, Utilities, and Mesh Tools	SVN (NE), SHARP SVN (MCS), XGitlab (CELS)	XGitLab (CELS)	Encrypted, externally available repository (TBD)
PROTEUS-NODAL	XGitLab (CELS), SVN (NE)		
PROTEUS-MOC	XGitLab (CELS), SHARP SVN (MCS)		
PROTEUS-SN	SHARP SVN (MCS)		
CSAPI	SVN (NE)		

PROTEUS should be re-organized by pulling all of the basic libraries, utilities, and mesh tools into a common location shared by the three solvers. Non-trivial effort is required to upgrade all of the current solvers to use the same basic libraries due to divergence and reliance on external packages (e.g. PETSc).

Access to the XGitLab repositories will be granted through a procedure requiring (1) an Argonne MCS account and active foreign visitor registration (for foreign nationals), (2) a valid PROTEUS license at the user's institution obtained through Argonne's technology and commercialization division, and (3) permission from the PROTEUS development team to access selected parts of the repository. Once an encrypted repository is set up, additional requirements may apply.

4.2 Unit and Regression Tests

Multiple unit and regression tests already exist for the components of PROTEUS. However, the number of tests for each solver should be expanded to test a larger percentage of the code. The unit and regression tests for each solver will be identified, documented, and expanded to increase code coverage. Creation of unit and regression tests will become an integral part of new feature development under a future SQA plan, such that each new subroutine added has a corresponding test.

4.3 Configuration Management

PROTEUS currently uses a manual pre-processing file to configure the compilation of the solvers (assigning external library locations and compiler options). This is not ideal for efficiently testing multiple compilation options or for recording compilation options for later reference. Improved configuration management is highly recommended to enable the user and automated testing server to compile via command line and record the configuration options. This will require non-trivial effort due to the lack of automated configuration and complexity involved in transitioning to tools like the GNU Build System. As a starting point, the existing Makefile workflow should be cleaned up and simplified as much as possible.

4.4 Automated Testing

A multi-core machine should be dedicated for the unit and regression testing of PROTEUS. One is already available but is being shared for testing of other software which limits its availability for continuous integration tests. Verification tests are planned to be launched automatically by an automated server. The tests will be either scheduled on a daily basis or launched based on repository commits (continuous integration), depending on which is more appropriate based upon the resource size and number of commits.

The PROTEUS-SN code and one of the mesh utilities is already tested automatically via the Buildbot server for two configurations on a nightly basis. All other codes are tested using scripts manually executed by each software developer before pushing the source to the repository. The manually tested codes need to be incorporated in the automated process. Deployment of continuous integration testing will be explored as opposed to scheduled testing. The Jenkins server or GitLab CI should also be explored as an alternative to Buildbot.

4.5 Software Development Life Cycle

The development life cycle should be clearly defined in terms of how bug fixes and new features are requested, performed, tested, and released to users. Figure 2 depicts a proposed software development cycle for PROTEUS. The flow chart is designed based on the “SCALE Development Process” [9], except the production code base and beta test community are eliminated due to a smaller set of users and development team for PROTEUS.

First, the code requirements and feature requests originate from the NEAMS Leadership Team and from external users of PROTEUS. This is important to ensure that the software meets the requirements of the users and the mission of the program. New feature requests should document the scope of work, timeframe, proposed testing strategy, documentation needs, sponsor/budget, and reviewer. The feature request must be approved by the designated project leader before work commences.

During the development work box, the developer(s) implement features in PROTEUS. Each source control “commit” is recommended to reference an approved development task. Tests must

be added to the test suite to verify the functionality of new features. Documentation should be updated as necessary.

Once the developer feels the code is ready for independent review, an independent technical reviewer thoroughly tests the new feature and reviews the associated documentation. The independent reviewer may run additional tests as desired, iterating back and forth with the development team until the code is deemed to run correctly. The project leader should sign off that the independent review has been completed successfully, and then the new feature can be ported to the production version of PROTEUS.

Once a new version of PROTEUS is released to users, they will invariably provide bug reports and feature requests, and the cycle begins again.

Figure 2. Proposed PROTEUS Software Development Life Cycle

4.6 Availability of Qualified Personnel and Resources

Compliance with a rigorous software quality assurance plan requires availability of qualified personnel, funding resources, and computing resources. A non-trivial portion of funding and personnel resources must be set aside each fiscal year for maintaining software quality. We estimate

that approximately 0.5 FTE is required for rigorous code maintenance, testing, and documentation of the PROTEUS tools listed in this report (approximately \$200K per year). If less funding is available, some software quality measures can certainly be achieved, but to a lesser degree. PROTEUS is currently in a period of intense development, but even in development lulls, some minimum level of resources should be provided in order to maintain the capability. The importance of providing ongoing support must be communicated to and appreciated by management who sets the technical priorities and funding levels.

5. Preliminary Software Quality Assurance Plan

Development of a software quality assurance plan for PROTEUS is necessary. The rigorous software quality assurance plan [9] and corresponding documents [10-13] in place for the SCALE package at Oak Ridge National Laboratory can be used as a model. The purpose of a software quality assurance plan is to provide the procedures, controls, and infrastructure to ensure that the end user and sponsors have a useful, reliable, accurate product.

Following the SCALE QA plan [9] as a model, the PROTEUS QA plan should outline, at a minimum:

- How documents will be controlled (manuals for using the code, procedures for modifying the code, bug reports, etc.)
- Individual roles and responsibilities including designation of the PROTEUS Project Leader, the Leadership Team (most likely the Reactor Product Line Lead in conjunction with the NEAMS Leadership Council), the Development Team, and Independent Reviewers, and QA Leader
- The overall process of PROTEUS development
 - The procedure for how features are requested
 - The procedure by which bugs are reported and fixed
 - Verification test expectations
 - The process by which the code is released
 - The process by which users are tracked (and which code version they have)

It is premature to detail all of these items without the infrastructure in place and without a commitment of ongoing support from the Leadership Team. The specification of a detailed quality assurance plan for PROTEUS is therefore postponed to a later date once support from management is obtained.

6. Next Steps

During FY19, bringing the codes up to the baseline requirements outlined in the previous sections should be the primary focus. Communication to the Leadership Team of the importance of ongoing support for SQA should be emphasized. Once the baseline requirements are fulfilled and support from leadership is obtained, a formal SQA process can be defined and put in place.

REFERENCES

1. PROTEUS: Simulation Toolset for High-Fidelity Reactor Physics Analysis, Website Address: <https://www.ne.anl.gov/codes/proteus/> .
2. C. H. Lee et al, "FY17 Status Report on NEAMS Neutronics Activities," ANL/NE-17/28, Argonne National Laboratory, September 30, 2017.
3. A. Marin-Lafleche, M. A. Smith, and C. H. Lee, "PROTEUS-MOC: A 3D Deterministic Solver Incorporating 2D Method of Characteristics," M&C 2013, Sun Valley, ID, May 5-9 (2013).
4. E. R. Shemon, M. A. Smith, and C. H. Lee, "PROTEUS-SN User Manual," ANL/NE-14/6 (Rev 3.0), Argonne National Laboratory, February 16, 2016.
5. M. A. Smith and E. R. Shemon, "User Manual for the PROTEUS Mesh Tools," ANL/NE-15/17 (Rev 2.0), Argonne National Laboratory, September 19, 2016.
6. C. H. Lee, A. Marin-Lafleche, and M. A. Smith, "Development of Cross Section Library and Application Programming Interface (API)," ANL/NE-13/15, Argonne National Laboratory, September 30, 2013.
7. C. H. Lee and W. S. Yang, "MC²-3: Multigroup Cross Section Generation for Fast Reactor Analysis," *Nuclear Science and Engineering*, **187**(3), 268-290 (2017).
8. M. A. Smith, E. E. Lewis, and E. Shemon, "DIF3D-VARIANT 11.0: A Decade of Updates," ANL/NE-14/1, Argonne National Laboratory, January 16, 2014.
9. B. T. Rearden, M. T. Sieger, S. M. Bowman, and J. P. Lefebvre, "Quality Assurance Plan for the SCALE Code System," SCALE-QAP-005, Rev. 4, Oak Ridge National Laboratory (2013).
10. B. T. Rearden, S. M. Bowman, and J. P. Lefebvre, "Configuration Management Plan for the SCALE Code System," SCALE-CMP-001, Rev. 8, Oak Ridge National Laboratory (2013).
11. B. T. Rearden, J. P. Lefebvre, and S. M. Bowman, "SCALE Procedure for Discrepancy Reports," SCALE-CMP-004, Rev. 5, Oak Ridge National Laboratory (2013).
12. W. J. Marshall and D. A. Reed, "SCALE Procedure for Verified, Archived Library of Inputs and Data (VALID)," SCALE-CMP-012, Rev. 2, Oak Ridge National Laboratory (2013).
13. B. T. Rearden and J. P. Lefebvre, "SCALE Procedure for Feature Changes," Oak Ridge National Laboratory (2013).

Nuclear Science and Engineering Division

Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439-4842

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC