NGEE Arctic Software Productivity and Sustainability Plan *

January 26, 2019

This Software Productivity and Sustainability Plan describes NGEE Arctic's process for developing and modifying software used in the project, consistent with CESD's software policy and procedures. This document is not a description of any technical work related to these tasks, but rather a description of the processes the team will use to accomplish that work. It takes into consideration that the software components used in this project originate outside the project and already have their own distribution protocols, development policies, and Software Productivity and Sustainability Plans.

The following software packages (hereafter collectively addressed as "software," "code," or "packages") may be modified as part of this work:

Advanced Terrestrial Simulator (ATS)	https://github.com/amanzi/ats
Alquimia	https://github.com/LBL-EESA/alquimia
PFLOTRAN	https://www.pflotran.org/documentation/index.html
FATES	https://github.com/NGEET/fates
E3SM	https://github.com/E3SM-Project/E3SM

These codes are open source and are developed independently of the NGEE Arctic project. Participants in the project will follow the existing development policies or Software Productivity and Sustainability Plans for those codes, to the extent practicable, and will augment those plans as needed.

Overall Process

The Test-Driven Development (TDD) software development process will be used throughout Phase 3. This process refers to a style of programming in which three activities are tightly interwoven: coding, testing, and design. TDD has a good track record for improving existing software and is well-suited for scientific software because it focuses on the need to maximize scientific output while ensuring software quality but avoiding unnecessarily formal methods for rigorously modeling and specifying the exact nature of the software products.

Development will be divided into phases, with each phase resulting in a new release. Within each phase, requirements will be determined directly through meetings with NGEE Arctic staff, the broader software development teams, and other interested parties. High-level requirements will then lead to specific test cases that will drive the development through multiple iterations of design and analysis, code development, and testing. The NGEE Arctic modeling team will use the project management tools from teamwork (www.teamwork.com) to communicate requirements and status of ongoing tasks.

Tools and Processes

The NGEE Arctic team will continue to use the existing software version control repositories at each code's hosting sites (i.e. BitBucket for PFLOTRAN, GitHub for the other codes). New development will be undertaken in separate feature branches. All new features will be accompanied by appropriate tests. The NGEE Arctic modeling team will use the project management tool teamwork (www.teamwork.com) to coordinate work internally and to track the ongoing tasks. Issues that affect the larger software communities will be tracked using the tools provided by each code's hosting site.

A pull request will be created to merge feature branches back into the main branch when development of that feature is complete. New stable versions will be distributed from the code hosting sites through tags.

Training

All new developers will be provided training material on scientific software development practices that was produced by the Interoperable Design of Extreme-scale Application Software (IDEAS) project. All new

developers will be assigned a mentor from the pool of existing developers for the purposes of practical exposure to the project development practices.

Software Improvement Strategies

The primary software improvement strategy for this project will be the institution of code reviews of the project software products. Code reviews will be conducted internally during each feature development phase to evaluate the overall design and testing strategy. In addition, informal code reviews with the larger software development teams will be undertaken as needed.

Risk Management for Third-Party Tools

ATS and PFLOTRAN rely on a significant number of open-source software dependencies. Those packages will be tested for correct functionality using their own integration tests and by running the suite of tests currently available for ATS and PFLOTRAN. To mitigate risk associated with new versions of software dependencies, the versions used before undergoing any migration to a newer version of the dependencies will be archived to provide a fallback plan. Furthermore, all migration to new major versions of dependencies will only be performed when new features are created for the ATS and PFLOTRAN software.

If the integration tests fail and the stack trace indicates that the failure was in a previously well-behaved dependency, then the development team will know that changes in that dependency have caused the error. In that event, the project team will (1) engage the third-party development team to see whether the loss of capability could be redeveloped or added back, or (2) seek to correct the inconsistency or add the missing capability on its own. In the case where the capability could not be recovered in its original distribution, the project team would seek to find the capability in another third-party dependency before attempting to develop a new piece of software to provide the capability.