Improving the Sustainability of SCEC Scientific Software using Software Best Practices

Philip J. Maechling[1]; Scott Callaghan[1], Kevin Milner[1], Edric Pauk[1], William Savran[1], Fabio Silva[1], Mei-Hui Su[1]

[1]Southern California Earthquake Center

SCEC uses **scientific software** to **deliver broad impact seismic hazard information** to the research community and the public. This year, our SCEC software sustainability efforts **applied best-practices** to SCEC software making it **easier to use** and **extend** in the future.

Software Development Best Practices for the CIG Community (Summary Table)

	Minimum	Standard	Target
Licensing	Open source	Same as Minimum	Same as Minimum
Version Control	All source in version control.	Differentiation between maintenance and new development.	(a) New features added in separate branches. (b) Stable development branches for rapid release of new features.
Coding		(a) User-friendly specification of parameters at run time. (b) Development plan, updated annually. (c) Comments in code with purpose of each function. (d) Users can add features or alternative implementations without modifying main branch. (e) User errors generate message that helps user correct the problem.	Standard + (a) Functionality implemented as a library rather than an application. (b) Output of provenance information. (c) Parallel access to inputs/outputs. (d) Checkpointing.
Portability, configuration, and building	(a) Codes builds on Unix-like machines with free tools. (b) Portable build system.	Minimum + (a) Dependency checking. (b) Automation and portability of configuration and building. (c) Each simulation outputs all configuration and build options for reproducibility.	Standard + (a) Selection of compilers, optimization, build flags during configuration without modifying files under version control. (b) Multiple builds using same source. (c) Allows installation to a central location.
Testing	(a) Code includes tests that verify it runs properly. (b) Results of accuracy and/or performance benchmarks (if established by the community).	Code includes pass/fail tests that verify it runs properly. (b) Development pipeline uses continuous integration to automate running test suite.	(a) Pass/fail unit testing for verification at a fine grain level. (b) Method of Manufactured Solutions for verification at a coarse grain level. (c) Use of code coverage tools to asses gaps in test coverage.
Documentation	(a) Instructions for installation. (b) Description of all parameters. (c) Explanation of physics the code simulates. (d) Cookbook examples with input files. (e) Citable publication. (f) Documentation provided online or offline.	(a) Description of workflow for research use. (b) Description of how to extend code in anticipated ways.	Standard + (a) Guidelines on parameter scales/combinations for which code is designed/tested. (b) FAQs or knowledge base. (c) Documentation provided in dynamic form and available offline.
User workflow		(a) Changing simulation parameters does not require rebuilding. (b) User-specified directories and filenames for input/output. (c) Use of standard binary formats. (d) Citation for code version.	Standard + Reproducibility via archiving of workflow.

Minimum: Practices that codes must follow in order to be accepted by CIG.

Standard: Practices in addition to the Minimum that should be used by all codes developed within the CIG community. Codes not meeting these standards should be actively working to eliminate deficiencies.

Target: Desirable practices beyond the Standard that developers should consider in defining development priorities for codes developed within the CIG community.

Figure 1: The primary objective of this project was to bring SCEC software distributions up to at the CIG minimum software best practices, as shown here on the right. The CIG minimum Software Development Best Practices guidelines mention 10 specific requirements, in several categories. We evaluated SCEC software distributions against these CIG practices, then made improvements to our software in order to meet these geoscience community software standards

Figure 2: We migrated the BBP development to a Linux server with current GNU compilers, increased unit test coverage to approx. 50%, and implemented CI processing for the BBP repository using GitHub actions. We improved the code and API documentation, reviewed and updated the scientific modules' documentation on GitHub. We updated the GitHub repository, to contain standardized files for our SCEC repositories, created bbp_docker images, and updated the software license to the USC recommended BSD-3 license.

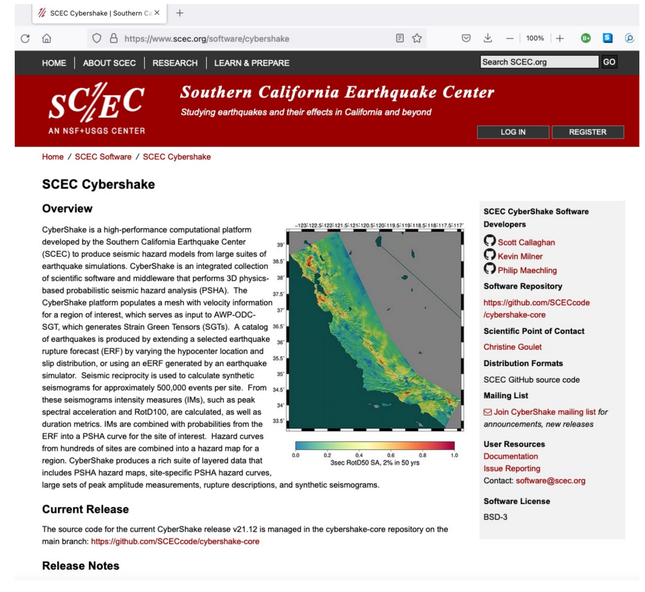

Figure 3: We migrated the CyberShake software system from CVM into three public SCEC GitHub repositories to reduce the complexity of the CyberShake repositories. We migrated the CyberShake job-submission host to a Linux server operated by USC CARC which significantly improves the reliability of the CyberShake workflow system. We updated the CyberShake software to use the most recent version of the BBP rupture, high-frequency wave propagation, and post-processing software. We created a standardized software page for the CyberShake project to act as a homepage for the project.

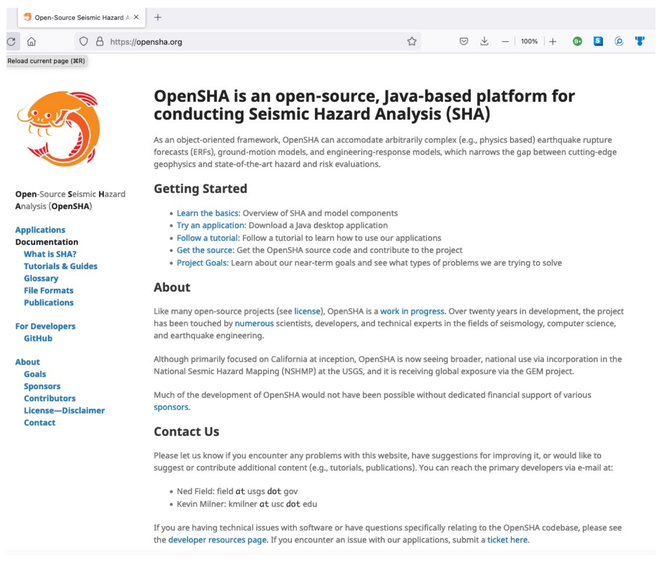

Figure 4: We re-organized the OpenSha GitHub repositories and archived several of the inactive OpenSHA GitHub repositories. We added Continuous Integration (CI) testing to the OpenSHA repo so selected tests are run on each software commit. We updated the OpenSHA license from the earlier Apache 2 license to the USC recommended BSD-3 to ensure un-restricted open-source access to all OpenSHA software in the future. We made improvements to the OpenSHA documentation which now provides overview materials, application examples, tutorials, and a development roadmap.

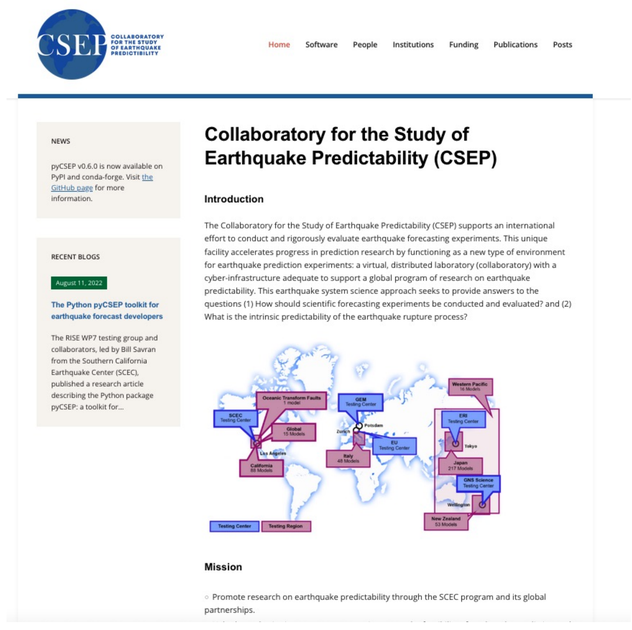

Figure 5: We implemented CI testing in the pyCSEP GitHub repository. We implemented pyCSEP documentation that describes the parameters used in each pyCSEP method. We released pyCSEP v0.6.0 through PyPi and and conda-forge. We developed Jupyter notebook-based pyCSEP software tutorials to provide interactive documentation to researchers. We developed a project website. We developed two citable software publications for the PyCSEP software including an article in the Journal of Open Source Software, and an article in Seismological Research Letters.

Figure 6: We reorganized the UCVM GitHub repos and updated the installation process to be more reliable. We implemented CI processing to run when changes are committed to the UCVM and ucvm_plotting repositories. We updated UCVM's GitHub wiki based UCVM documentation. We created UCVM DockerHub images containing individual velocity models. We registered the UCVM software to Zenodo and obtained a software DOI for the latest UCVM v22.7.0 release.