CSEP Software Development: Status, Priorities, Practices

Masha Liukis, Danijel Schorlemmer, John Yu, Philip Maechling, Jeremy Zechar, Max Werner, Fabian Euchner, Matt Gerstenberger, David Rhoades, Kazu Nanjo, Thomas Jordan

June 6, 2012
Rancho Mirage, CA, USA
All CSEP Testing Centers (228 models)
Model Submission Requirements

- http://northridge.usc.edu/trac/csep/wiki/CSEPModelRequirements

**CSEP SCEC Testing Center requirements for a model submission**

For the CSEP Testing Center at SCEC to support more forecasts groups and to host more models and algorithms, we request that modellers comply with the following set of requirements to enter their models into the Testing Center.

**Submission deadline**

The CSEP Testing Center at SCEC follows quarterly releases of January 1, April 1, July 1 and October 1 of each year. Source code repository is “frozen” one month prior to the release date meaning that there are no code changes allowed one month before the release date. For example, code freeze for January 1st release is December 1st of previous year (March 1st for the April 1st release, June 1st for the July 1st release, and September 1st for the October 1st release). This good software practice guarantees that all code changes targeted for upcoming release, including new models, are intensively tested for the whole month on SCEC certification server before code base of new release is installed on SCEC operational server. We ask modelers to allow enough time for their model submission before the code freeze for targeted CSEP release they would like their model to be part of. Installation of new model within testing center requires work on CSEP development team side after the model codes are submitted for installation. Please consider submitting the model at least two months prior to the targeted CSEP release date. This will allow for one month of interactive installation process between the modeler and the CSEP development team before the code freeze takes place.

**Installation**

**CSEPReq1** We request that the modeler provide source code along with rules to generate executable file if applicable. This includes specification of:

- All dependent packages the model is relying on (including version numbers), and rules for how to build them
- Compiler (including version number) if applicable
- Rules to build executable(s) is applicable

**CSEPReq2** We request that the modeler install the model’s source code on CSEP integration server csep-devel.usc.edu (accessible through motion.usc.edu), and builds executable(s) if applicable.

- Please contact John Yu (johnyu@...) to get an account

**Execution**

**CSEPReq3** We request that the modeler provide command(s) required to invoke the model.

Currently, CSEP testing framework supports ANSS and CMT authorized data sources.

- ANSS input catalog is provided in ZMAP format (ASCII or Matlab):
  1. Please refer to examples of input catalogs

- CMT input catalog is provided as a reduced data set of raw CMT catalog in ASCII format
  1. Please refer to examples of input catalog
System Design and Implementation Principles

- Use object-oriented design.
- Only add features when they are needed
  - Testing Framework is released on a regular 3-month basis
- Use open-source software whenever possible.
System Design and Implementation Principles

- Use object-oriented design.
- Only add features when they are needed.
- Use open-source software whenever possible.
- Put the system under test to ensure reliability and to make the system easier to change.
System Design and Implementation Principles

- Use object-oriented design.
- Only add features when they are needed.
- Use open-source software whenever possible.
- Put the system under test to ensure reliability and to make the system easier to change.
- Use YUM repository and KVM virtual images for software distribution.
System Design and Implementation Principles

- Use object-oriented design.
- Only add features when they are needed.
- Use open-source software whenever possible.
- Put the system under test to ensure reliability and to make the system easier to change.
- Use YUM repository and KVM virtual images for software distribution.
- CSEP testing center codes are open source.  
  – cseptesting.org/software
System Design and Implementation Principles

- Use object-oriented design.
- Only add features when they are needed.
- Use open-source software whenever possible.
- Put the system under test to ensure reliability and to make the system easier to change.
- Use YUM repository and KVM virtual images for software distribution.
- CSEP testing center codes are open source.
- Provide communication tools
  - email-lists (csep-all.intensity.usc.edu; info@cseptesting.org)
  - Trac (http://northridge.usc.edu/trac/csep)
CSEP Recent Development

- **Generic Functionality**
  - Support for polygon-based forecasts
  - T (the classical paired t-test) and W (the Wilcoxon signed-rank) evaluation tests introduced by David Rhoades, GNS Science, NZ
  - KVM virtualization to deploy and distribute CSEP software
  - SVN archive to store raw catalogs as downloaded from authorized data sources (ANSS, CMT) (CSEP V12.4.0)
CSEP Recent Development

- SCEC Testing Center at USC
  - Oceanic Transform Faults experiment
    - Oceanic Transform Fault 1-day model by Margaret Boettcher
  - 2 EAS (EEPAS allowing for AfterShocks) 3-month models by David Rhoades for California testing region
  - SHIFT_GSRM 1-year model by Peter Bird for high-resolution Global testing region
    - SHIFT = Seismic Hazard Inferred From Tectonics (a set of hypotheses/rules from Bird & Liu [2007, SRL]); and
    - GSRM = Global Strain Rate Map (a global dataset from Kreemer et al. [2003, GJI], and a major project of the International Lithosphere Project)
CSEP Future Development

- Evaluation tests to support polygon-based forecasts.
- Shorten testing periods from 1 day to either shorter intervals or to event-based testing to capture short-term fluctuations.
- Support cross forecast class evaluation.
- Testing Centers results viewer.
CSEP Testing Framework

- Earthquake Early Warning (EEW)
CSEP Testing Framework

- Transient Detection