"OpenQuake: Calculate, share, explore"

Testing procedures adopted in the development of the risk component of the OpenQuake-engine

Authors
Anirudh Rao[1]

[1] GEM Model Facility
via Ferrata, 1
20133 Pavia
Italy

Email address (for all the authors):
<name.surname>@globalquakemodel.org

Citation
Please cite this document as:
Rao, Anirudh (2015). Testing procedures adopted in the development of the risk component of the OpenQuake-engine *Global Earthquake Model (GEM) Technical Report 2015-04.*
doi: 10.13117/GEM.OPENQUAKE.TR2015.03, 27 pages.

Third printing, March 2015

Contents

Part I

Introduction

1. Software Testing

The current document describes the testing procedures adopted in the development of the hazard component of the OpenQuake-engine (oq-engine), the open source hazard and risk software developed by the Global Earthquake Model initiative.

Nowadays seismic hazard analysis serves different needs coming from a variety of users and applications. These may encompass engineering design, assessment of earthquake risk to portfolios of assets within the insurance and reinsurance sectors, engineering seismological research, and effective mitigation via public policy in the form of urban zoning and building design code formulation.

Decisions based on seismic risk results may have impacts on population, properties and capitals, possibly with important repercussions on our day-to-day life. For these reasons, it is recommendable that the generation of hazard models and their calculation is based on well-recognized, state-of-the-art and tested techniques, requirements that must be reconciled with the need to regularly incorporate recent advances given the progress carried out within the scientific community. The features described below contribute to fulfill these requirements:

- Software should have a modular and flexible structure capable of incorporating new features and - as a consequence - offer users the most recent and advanced techniques. In very general terms, modularity is the level to which a component of a system can be moved, replaced or reused. In software design, modularity means the separation of the software into smaller independent components that can be implemented, maintained and tested easily and efficiently.
- Software should have and extensive test coverage which captures possible errors and avoids regressions (i.e. unexpected behaviors introduced by new features). Software testing (Myers et al., 2012) is an important, complex and vast discipline which helps in developing methods and processes aimed at certifying the extent to which a computer code behaves according to the original design intent and user specifications.

The oq-engine includes different levels of modularity. The first is the one separating the engine itself into a number of libraries (see Figure 1.1), each one containing well identified knowledge, objects and methods (e.g. the OQ-hazardlib includes objects and methods needed to compute probabilistic seismic hazard and the OQ-risklib contains methods to compute scenario and probabilistic seismic risk). The second one pertains to the data model adopted in the development of each library as a result of the abstraction process. According to Berkes (2012) scientific

./Pictures/qa/oq-engine_structure.pdf

Figure 1.1 – *A schematic describing the main components of the OpenQuake-engine software.*

software must be:

- Error proof
- Flexible and able to accommodate different methods
- Reproducible and re-usable

1.1 Testing and Quality Assurance

Despite the distinction between software testing (in some cases also called Quality Control) and being somewhat vague and partly open to personal judgment, it's clear that is a more comprehensive and overarching process than software testing. aims at the definition of the best processes that should be used to provide guarantees that user expectations will be met. Software testing focuses instead on detecting software faults by inspecting and testing the product at different stages of development.

1.1.1 Testing

Software testing can be implemented at different stages of the development process, with varying strategies to approach the problem. The oq-engine and the associated libraries are developed following an agile paradigm. This development strategy is organized in a way that the creation of the real code is completed in parallel and fully integrated with the software testing process.

The software engineering community provides a wide range of testing levels and typologies. In the current document we consider just a portion of them with the specific intent to illustrate the standards used in the development of the oq-engine and particularly of its hazard component.

1.1.2 Quality Assurance

From the IEEE "Standard for Software Quality Assurance Processes": *Software quality assurance is a set of activities that define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software* *products of suitable quality for their intended purposes. A key attribute of SQA is the objectivity of the SQA function with respect to the project. The SQA function may also be organizationally independent of the project; that is, free from technical, managerial, and financial pressures from the project.* In this document we are not covering topics related to since this would go beyond its scopes.

1.2 Document structure

The document is organized into four main chapters and two appendixes.

In the current chapter we provide a very brief and general introduction to software testing with a focus on the testing of scientific software.

In the second chapter we describe the module, or unit, testing, the acceptance tests adopted in the development of the oq-engine and we discuss some examples.

In the third and fourth chapters we illustrate tests comparing the results computed with the oq-engine against the ones computed using different probabilistic seismic hazard analysis software.

Appendix **??** provides details on the PEER tests implemented in the .

Part II

Unit Tests

2. Unit Testing in the OpenQuake-engine

Part III

Acceptance Tests

3. Framework for Acceptance Testing

4. Acceptance Tests for the Scenario Risk Co

5. Acceptance Tests for the Classical Risk Co

6. Acceptance Tests for the Event-Based Risk

Part IV

Benchmark Tests

7. Comparing Loss Curves from the Classical

8. Comparison with Other Softwares

Part V

Performance Tests

9. Demos

Books
Articles
Reports

10. Stress Tests

Bibliography

Books

Myers, G. J., C. Sandler, and T. Badgett (2012). *The art of software testing*. Wiley and Sons, Inc. (cited on page 7).

Articles

Other Sources

Berkes, P. (2012). *Writing robust scientific code with testing (and Python)*. Euroscipy. URL: http://archive.euroscipy.org/talk/6634 (cited on page 7).