INL/EXT-14-33985
Revision 0

Light Water Reactor Sustainability Program

Software infrastructure progress in the RAVEN code

December 2014

DOE Office of Nuclear Energy

INL/EXT-14-33985
Revision 0

Light Water Reactor Sustainability Program

Software infrastructure progress in the RAVEN code

Joshua J. Cogliati
Cristian Rabiti
Cody J. Permann

December 2014

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov/lwrs

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

EXECUTIVE SUMMARY

The milestone M4LW-15IN0704043 has been achieved. RAVEN has been migrated to Gitlab which adds new abilities for code review and management. In addition, a standalone RAVEN framework packages have been created for OSX and two Linux distributions.

CONTENTS

FIGURES

ACRONYMS

RAVEN	Risk Analysis Virtual Environment
MOOSE	Multiphysics Object-Oriented Simulation Environment

Software infrastructure progress in the RAVEN code

1. Creation of RAVEN Binary Packages

Prior to the work on milestone M4LW-15IN0704043, the only fully supported ways of installing the RAVEN[1,2] package involved access to the source code and compiling it. While this is optimal for developers, it is more complicated for future users of the software who are only planning on running RAVEN. The goal of this milestone is to report how this issue has been addressed by introducing new installation methods which are more transparent to the final users. RAVEN uses portions of MOOSE[3,4] for its testing framework and for reading and writing MOOSE input files.

To accomplish simplifying the installation of the RAVEN framework, three binary installation packages and a source installation package have been created. For Linux, on current versions of Fedora (version 20) and Ubuntu (version 14.04) all the necessary external dependencies (that is, software packages that are used by RAVEN but not developed by the project) are available in the Linux distribution, so a yum (for Fedora) and apt-get (for Ubuntu) command is provided, which will install the dependencies. For OSX Mavericks, not all the dependencies are available as part of operating system, so a .dmg file with a package inside is provided to install these. The following external dependencies are compiled into the OSX package:

- Perl Compatible Regular Expressions 8.35
- SWIG 2.0.12
- Libtool 2.4
- Setuptools 2.1
- BLAS and LAPACK 3.4.2
- Numpy 1.7.0
- HDF5 1.8.12
- Cython 0.18
- H5py 2.2.1
- Scipy 0.12.0
- Scikit learn 0.14.1
- Freetype 2.4.12
- Libpng 1.6.12
- Matplotlib 1.4.0

The Linux packages are tarballs that can be unpacked and run once the needed packages from the distribution are installed. The OSX .dmg file includes both the external dependencies and the RAVEN framework. For all three binary packages, a compiled version of the CROW package is included. CROW is a Python library developed by INL to provide probability distributions and other tools used by RAVEN.

The new packages only include the portions of MOOSE that are required by the RAVEN framework, which decreases the size of the packages. The RAVEN packages do not need the entire MOOSE system environment to be setup.

2. Migrate RAVEN to Gitlab

The RAVEN code has migrated from its existing Subversion[5] repository to INL's HPC gitlab[6] repository. Subversion and git[7] are revision control systems that allow different versions of the software to be stored. Subversion is a centralized version control system with limited ability to modify the version history once it is committed to the repository. Git is a distributed version control system, and it allows the versions of the software to be transferred around and modified in multiple ways. Gitlab is a software environment that helps manage a git repository via a web interface. While the migration from subversion to gitlab adds complexity, it brings advantages. Subversion supports branches[a], but git is much more flexible with branches, and the configuration of the previous MOOSE subversion repository prevented subversion branches from being used. The new RAVEN gitlab repository fully supports branches. This allows improved support for independent code branches and to review code before it is joined to the main development branch. The added ability to review the new code development before it is committed has already resulted in an increase of constructive interaction between the developers which has resulted in improved code. The branches will allow release branches to be created that only have stable updates used.

3. Installation Instructions

3.1 Framework Source Install

The first step is the installation of the dependencies. Either they can be installed manually or the script can be used (other versions of the following libraries might work too).

1. numpy-1.7.0
2. hdf5-1.8.12
3. Cython-0.18
4. h5py-2.2.1
5. scipy-0.12.0
6. scikit-learn-0.14.1
7. matplotlib-1.4.0

Second, the user needs to untar the source install (if there is more than one version of the source tarball, the full filename will need to be used instead of *):

```
tar -xvzf raven_framework_*_source.tar.gz
```

If the dependencies were not installed previously, they can be installed by running the `raven_libs_script.sh`:

```
cd trunk/raven/
./raven_libs_script.sh
cd ../../
```

The next step is the compilation of CROW. This can be done either with the python setup or with a makefile. Using the setup.py file:

```
cd trunk/crow/
python setup.py build_ext build install --user
```

[a] Branches are parallel lines of code development tracked by the version control system. For example, a developer could work on a feature on one branch while other developers worked on the main line of development, and then when the feature was done it could be merged with the main line of development. Another use of branches is to have a release branch, that only fully stable features are added to, and a development branch that more experimental features are added to.

Now the make file can be run to compile the code:

```
cd trunk/crow/
make -f Makefile.linux
```

Finally the tests to verify the proper installation of RAVEN can be used (it might be needed to change the `cd` command if not in the crow directory):

```
cd ../raven/
./run_tests --re=framework --skip-config-check
```

The user could expect a printing similar to: `8 passed, 19 skipped, 0 pending, 0 failed` at the end. Unless the number of failed tests is greater than zero the testing should be considered successful. The large number of skipped tests is usually due to the fact that the regression testing checks the presence of other codes that are usually used in conjunction with RAVEN. If those codes are not presents the regression test script detects it and skips the corresponding tests

3.2 Ubuntu Framework Install

The installation sequence is described below:

Install the dependencies:

```
sudo apt-get install libtool python-dev swig g++ python3-dev \
 python-numpy python-sklearn python-h5py
```

Untar the binary install (if there is more than one version of the binary install, the full filename will need to be used instead of *):

```
tar -xvzf raven_framework_*_ubuntu.tar.gz
```

Run the tests:

```
cd trunk/raven/
./run_tests --re=framework --skip-config-check
```

There should be a line like: `8 passed, 19 skipped, 0 pending, 0 failed` at the end. If any failed, look at the output to see why.

3.3 Fedora Framework Install

The installation sequence is described below:

Install the dependencies:

```
yum install swig libtool gcc-c++ python-devel python3-devel \
 numpy h5py scipy python-scikit-learn python-matplotlib-qt4
```

To make it possible to be able to edit and rebuild the manual, install:

```
yum install texlive texlive-subfigure texlive-stmaryrd
```

Untar the binary install (if there is more than one version of the binary install, the full filename will need to be used instead of *):

```
tar -xvzf raven_framework_*_fedora.tar.gz
```

Run the tests:

```
cd trunk/raven/
```

```
./run_tests --re=framework --skip-config-check
```

There should be a line like: `8 passed, 19 skipped, 0 pending, 0 failed` at the end. If any failed, look at the output to see why.

3.4 OSX Framework Install

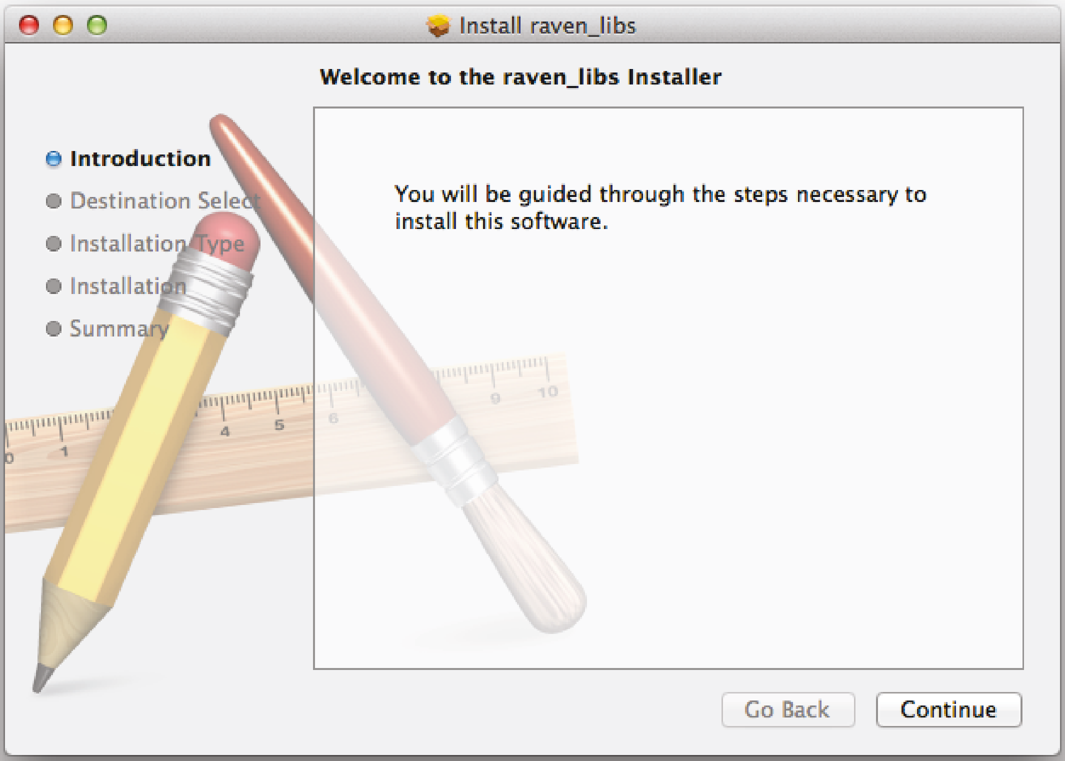
The installation sequence is described below:

Install XCode command line tools from Apple.

Open up the file `raven_libs_framework_and_crow.dmg`

Open up the `raven_libs.pkg` inside, and install it.

The files will be installed into `/opt/raven_libs`. The installer will edit the `.bash_profile` of the user who installs the package to source the `/opt/raven_libs/environments/raven_libs_profile` file. This file sets up the `PYTHONPATH` and the `PATH` so that the `raven_framework` command can be used.



Welcome to the raven_libs Installer

You will be guided through the steps necessary to install this software.

Figure 1: RAVEN OSX installer.

3.5 MOOSE and RAVEN Source Install

First, MOOSE should be installed. Follow the instructions for MOOSE:
```
http://mooseframework.org/getting-started/
```
Next, if the C++ RAVEN is desired, RELAP-7 needs to be installed. Follow the RELAP-7 instructions, but MOOSE needs to be installed in the same directory level as RELAP-7.

Then clone CROW and RAVEN:
```
git clone git@hpcgitlab.inl.gov:idaholab/crow.git
git clone git@hpcgitlab.inl.gov:idaholab/raven.git
```

Install the RAVEN dependencies via one of the methods mentioned for the RAVEN framework.

Then compile RAVEN:
```
cd raven
make
```

Then run the tests:
```
./run_tests
```

There will be a line telling how many passed. If any failed, look at the output to see why.

4. Conclusions

The milestone M4LW-15IN0704043 has been achieved. For code developers, the migration to Gitlab adds new abilities code review and code management that has already resulted in an increase in constructive interaction between the developers resulting in improved code. For non-developers, the new binary packages will make RAVEN easier to install. The tasks for this milestone improve things for both developers and users.

5. References:

[1] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, and R. Kinoshita, “RAVEN, a New Software for Dynamic Risk Analysis”, in Proceedings for PSAM 12 Conference, Honolulu (USA), 2014.

[2] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, and A. Naviglio, “RAVEN and dynamic probabilistic risk assessment: Software overview,” in Proceedings of ESREL European Safety and Reliability Conference (2014).

[3] D. Gaston, G. Hansen, S. Kadioglu, D. Knoll, C. Newman, H. Park, C. Permann, and W. Taitano, “Parallel multiphysics algorithms and software for computational nuclear engineering.” Journal of Physics: Conference Series, 180(1)(2009), 012012.

[4] http://www.mooseframework.org

[5] http://subversion.apache.org

[6] https://about.gitlab.com

[7] http://git-scm.com