Introduction to GitHub and SU2 Development Practices

SU2 Winter Workshop
February 3rd, 2017

Dr. Thomas D. Economon
Department of Aeronautics & Astronautics
Stanford University

SU2
The Open-Source CFD Code
It’s bright where we’re headed.

How do we get there? Scalable development practices.

- How do we avoid code conflicts?
  - Branching model in git for decentralized, parallel development.

- How does one contribute code contributions to the repo?
  - Pull requests through GitHub.

- Quality assurance?
  - Automatic, pre-merge regression testing (Travis CI) and code reviews.

- How do we minimize the overhead of software development in a research environment?
  - All of the above + streamlined release process at regular, frequent intervals.
1. Decentralized development in git

source: http://xkcd.com/1597/
1. Each new feature/capability should have its own branch. Note: internal devs should create branches directly in SU2 repo (not forks) to increase collaboration.

2. All branches operate in parallel, with “owners” updating their feature branches from develop regularly, i.e., ‘$ git merge develop’.

3. Once ready, owners prepare a pull request for feature. Code is reviewed, and after tests pass, merged into develop. Remove feature branch.

4. At regular intervals, develop is staged for a release. Once ready, it is pushed to master, tagged, and released. Note: master is always stable.
1. Decentralized development in git

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
<th>Updated by</th>
<th>Days Ago</th>
<th>Pull Requests</th>
<th>Merged?</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>Updated 13 days ago by economon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_home</td>
<td>Updated 10 hours ago by vdweide</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>develop</td>
<td>Updated a day ago by fpalacios</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_renaming</td>
<td>Updated 2 days ago by fpalacios</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_pywrapper</td>
<td>Updated 5 days ago by tobadavid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_DE_AD</td>
<td>Updated 6 days ago by rsanfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_driver_prototype</td>
<td>Updated 12 days ago by LaSerpe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_DE_AD_ALE</td>
<td>Updated 17 days ago by rsanfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_output</td>
<td>Updated 19 days ago by economon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_nlopt</td>
<td>Updated 19 days ago by talbring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_incompressible</td>
<td>Updated 21 days ago by talbring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_DE</td>
<td>Updated a month ago by rsanfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_FSI_FEA</td>
<td>Updated a month ago by rsanfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>feature_HB</td>
<td>Updated a month ago by arubino</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fix_addGlobalElementIndex</td>
<td>Updated a month ago by rsanfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Submitting code to the repository

- Submit contributions through pull requests on GitHub.
- Pull requests should target the develop branch.
- Both internal (internal branches) and external developers (external forks).

Reasons for pull request method:
- Keeps team informed (emails, PR description, commit logs).
- Allows for code review (GitHub).
- Automatic, pre-merge testing (Travis CI).
3. Continuous integration

- Pull requests are automatically tested against our suite of regression tests… we know upfront if there are problems and won’t merge!
- New features should also include new tests to ensure that the functionality is protected long-term.
- The develop branch is frequently tested automatically, but folks can activate for their own branches while they develop (and change notification to just their own email).
4. Releases

- We put out releases at frequent intervals:

  SU2 vX.Y.Z where X = major, Y = minor, Z = maintenance

- Released through GitHub (tags) and binaries are created and posted for download on su2.stanford.edu.

- Release schedule is dictated by a combination of features, events (e.g., AIAA for impact), and maintenance needs.

- Feature “hiding” is a practice we use to stage developments and get some early testing for features that aren’t ready for public consumption.
4. Releases

**SU2 version 5.0.0 "Raven"**

@economon released this 15 days ago

SU2 v5.0.0 contains major new features and improvements, such as the following:

- New in-memory Python wrapping of SU2 using SWIG with accompanying high-level API.
- Class enhancements for multiphysics applications, including interpolation and transfer.
- Free-form deformation (FFD) extensions, including bezier curves and improved usability.
- Reorganization of the incompressible solver for future expansion.
- Harmonic Balance flow analysis capability.
- Algebraic transition model implementation.
- More and better boundary conditions (accuracy and convergence improvements).
- Extensions to scripting for automated database creation (compute_polar.py).
- Critical improvements in I/O, including more feedback to the user.
- Additional bug fixes, stability improvements, and general code maintenance.

The following binary versions are available for download (serial only):

- macOS Sierra 10.12.2: Apple LLVM version 8.0.0 (clang-800.0.38)
- Linux (Redhat 7.0): g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-4)
- Linux (Ubuntu 16.04): g++ (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609

Download the binaries, source code, and test cases from the SU2 download page: http://su2.stanford.edu/download.html

**Downloads**

- Source code (zip)
- Source code (tar.gz)