Hall C Software Development

From the perspective of a user

Outline

- Integration of Hall A/C software: Version management with Git
  - Overview of git in practice
  - Examples of pushing/pulling updates
  - Issue tracking
- Configuration of Hall A/C software: Development of a new build system – SCons
  - Overview of SCons
  - Current status of build/configure system

Hall A/C Coordinated Development with Git

- In response to previous recommendation, we are now managing Hall A and Hall C software efforts with git.
- Hall C analysis (hcana) is being developed within the Hall A framework (PODD) -> having an effective development management system is important to coordinate these efforts.
- Hall C development has already spurred new efforts to improve PODD.
- Ongoing Fortran/hcana comparisons in parallel with new development makes version control/management crucial to ensure meaningful comparisons.
- Modular nature of experiments in Hall C (and Hall A) requires ongoing and continuous development of new analysis software, on an experiment-by-experiment basis -> version control is essential

Git Advantages (User Perspective)

- Documentation of procedures to download and install software exists, and is continually updated (Wiki)
- Git (in contrast to CVS, Subversion, etc.) is based on filesystem "snapshots", and not "deltas"-> much easier to back out changes … this makes the development effort significantly easier for new users.

Development Workflow using Github.com

- Online project hosting using git … many practical features which allow easy visualization of development path.
- Developers use their own account on github.com to create "forks" of JeffersonLab repositories.
- Changes are committed to a developer's forked repository, and at some point may be merged with the main JeffersonLab repository, via a pull request.
- The "gatekeeper" for the Jefferson Lab repository governs this process (Ole Hansen in Hall A and Steve Wood in Hall C).
- Gatekeeper model maintains limited access by developers/users to Jefferson Lab repositories -> important for comparison studies
- Single gatekeeper model may evolve as we get closer to running

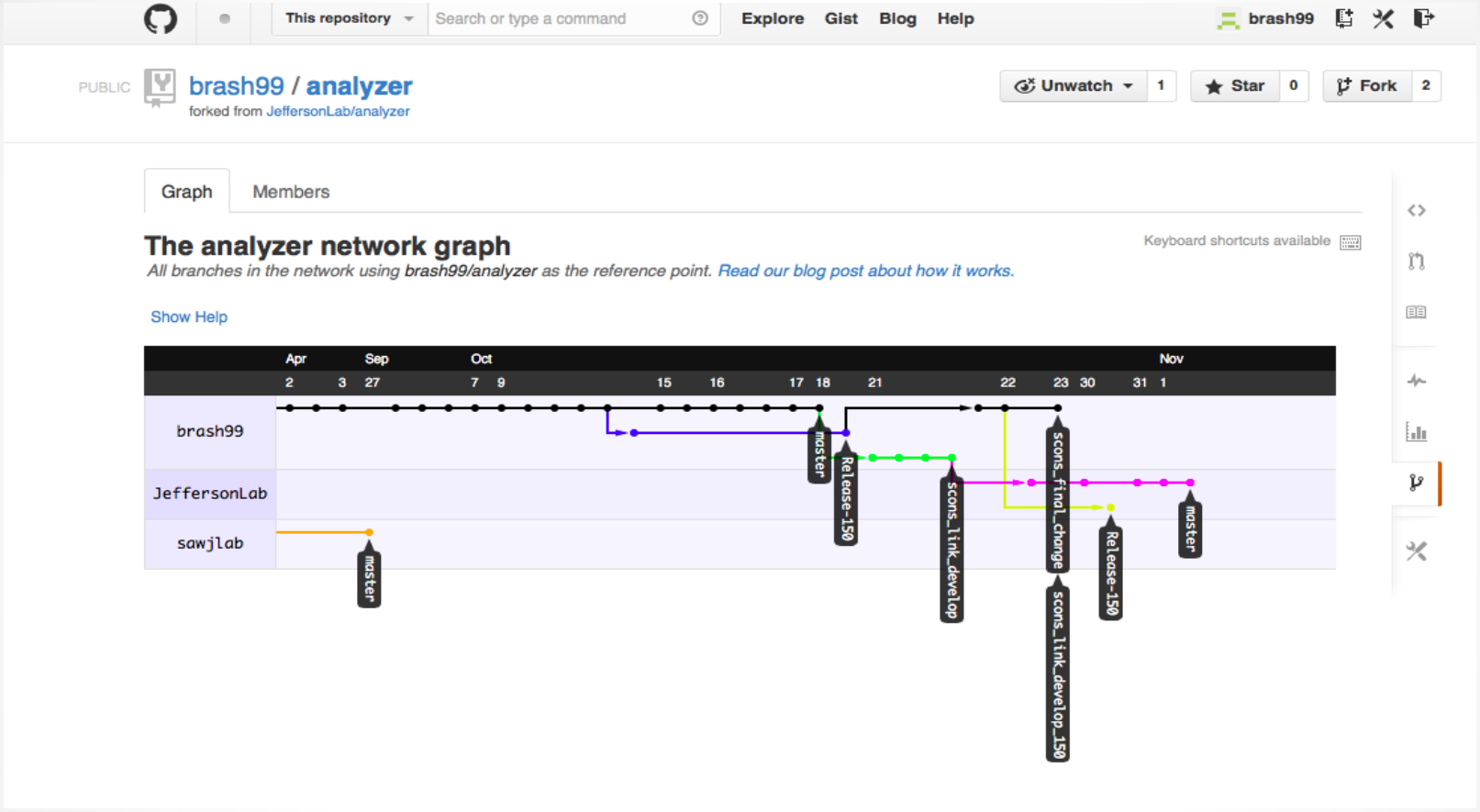
Current JLab Git Projects

- Hall A C++ Analyzer (Hall C submodule)
- Hall C C++ Analyzer
- Hall C Fortran engine
- Hall C Replay x 2 (for comparisons)
- Hall C Geant simulation of Compton Polarimeter
- SIMC (Monte Carlo for Halls A and C)
- SHMS Monte Carlo (for proposal development)
- TreeSearch (Hall A TreeSearch track reconstruction)

Development under Git

- Easy to install and update code on JLab systems as well as on local (Mac and Linux) machines
  - git clone git@github.com:brash99/analyzer.git
  - git branch –a
    - Master
    - remotes/origin/HEAD -> origin/master
    - remotes/origin/Release-070
    - remotes/origin/Release-100
    - remotes/origin/Release-110
    - remotes/origin/Release-120
    - remotes/origin/Release-130
    - remotes/origin/Release-140
    - remotes/origin/Release-150
    - remotes/origin/master
    - remotes/origin/scons_final_change
    - remotes/origin/scons_link_develop
    - remotes/origin/scons_link_develop_150
  - git pull origin <branch-name>

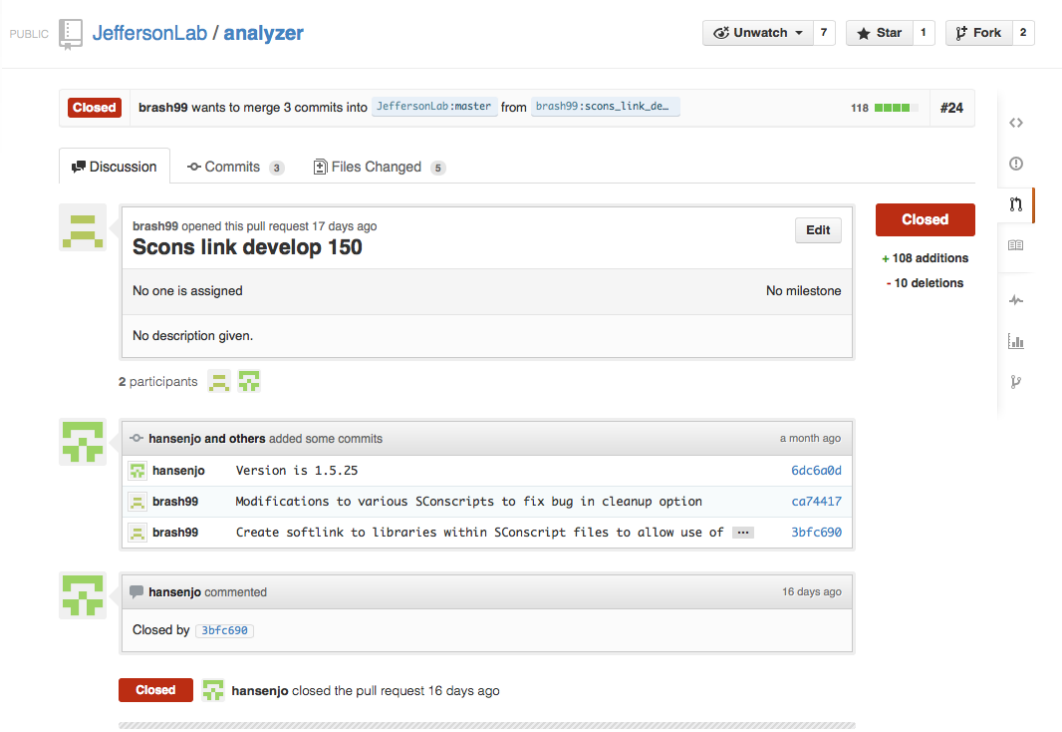
Example Network Graph



$ git checkout –b scons_final_change
(make changes)
$ git commit –a
$ git push origin scons_final_change

[Analyzer Network]

Pull Requests



Users/developers who are "watching" development of the JeffersonLab/analyzer repository are notified by email of pull requests.

Typically, other developers, as well as the gatekeeper, can make comments on the proposed pull request prior to it being accepted (or rejected).

Details of Pull Request

PUBLIC brash99 / analyzer
forked from JeffersonLab/analyzer

Unwatch 1 | Star 0 | Fork 2

Create softlink to libraries within SConscript files to allow use of system-wide SCons installation. Browse code

scons_final_change + scons_link_develop_150

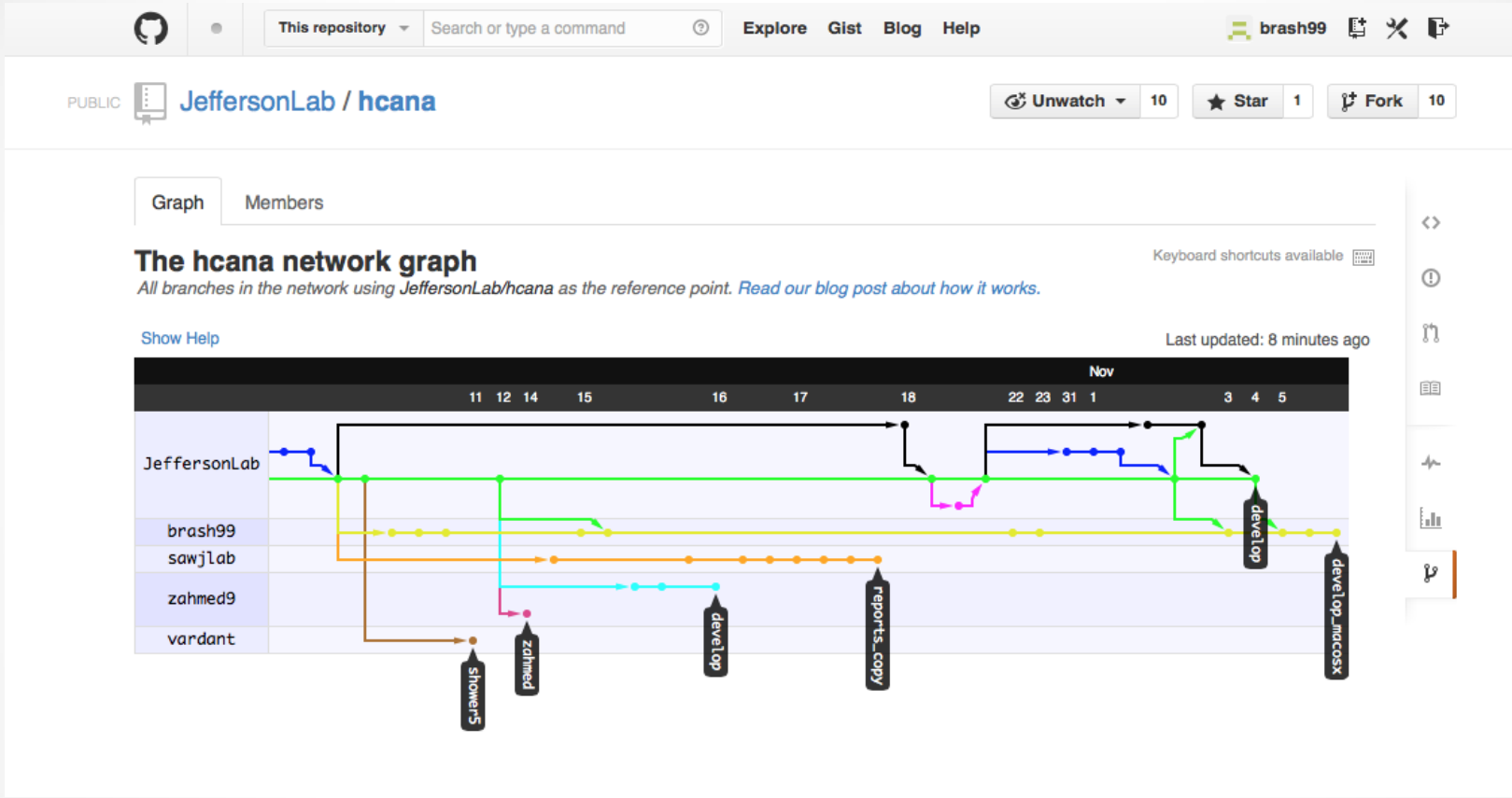
brash99 authored 17 days ago — 1 parent ca74417 commit 3bfc690f7e7ead6b60ee1decb02e65db2ab3e465

Showing **4 changed files** with **103 additions** and **38 deletions**. Show Diff Stats

12 SConstruct.py — View file @ 3bfc690

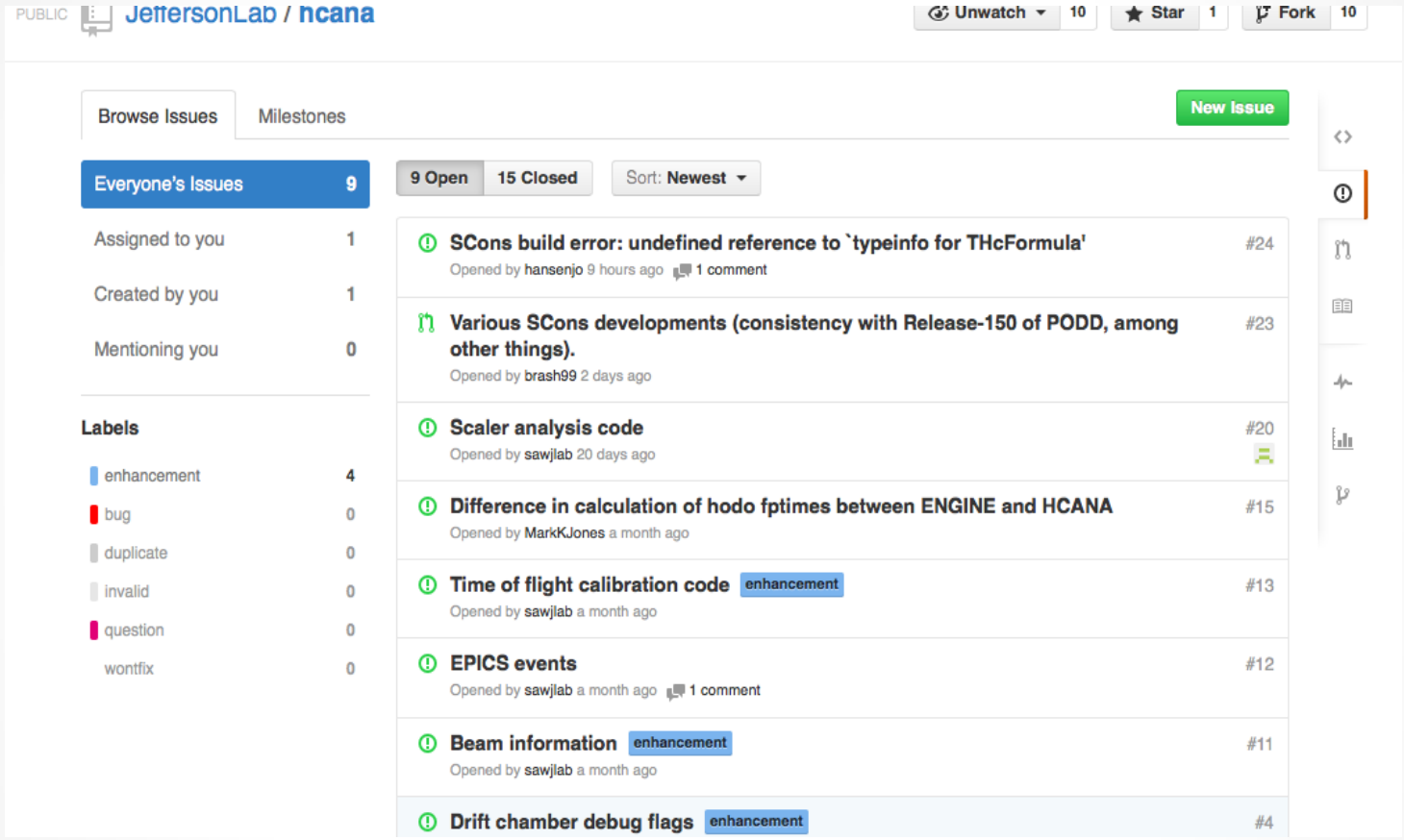
```
...  ...  @@ -30,9 +30,11 @@ def rootcint(target,source,env):
30   30    #  print "Construction variable = '%s', value = '%s'" % (key, dict[key])
31   31
32   32    ####### Check SCons version ##################
33        -print('!!! You should be using the local version of SCons, invoked with:')
34        -print('!!! ./scons/scons.py')
35        -EnsureSConsVersion(4,9,9)
     33   +#print('!!! You should be using the local version of SCons, invoked with:')
     34   +#print('!!! ./scons/scons.py')
     35   +print('!!! Building the Hall A analyzer and libraries with SCons requires')
     36   +print('!!! SCons version 2.1.0 or newer.')
     37   +EnsureSConsVersion(2,1,0)
36   38
37   39    ####### Hall A Build Environment #############
38   40    #
...  ...  @@ -41,8 +43,10 @@ def rootcint(target,source,env):
41   43    baseenv.Append(HA_SRC = baseenv.subst('$HA_DIR')+'/src ')
42   44    baseenv.Append(HA_DC = baseenv.subst('$HA_DIR')+'/hana_decode ')
43   45    baseenv.Append(HA_SCALER = baseenv.subst('$HA_DIR')+'/hana_scaler ')
44        -baseenv.Append(SOVERSION = '1.5')
     46   +baseenv.Append(MAJORVERSION = '1')
     47   +baseenv.Append(MINORVERSION = '5')
45   48    baseenv.Append(PATCH = '25')
     49   +baseenv.Append(SOVERSION = baseenv.subst('$MAJORVERSION')+'.'+baseenv.subst('$MINORVERSION'))
46   50    baseenv.Append(VERSION = baseenv.subst('$SOVERSION')+'.'+baseenv.subst('$PATCH'))
47   51    baseenv.Append(EXTVERS = '')
48   52    baseenv.Append(HA_VERSION = baseenv.subst('$VERSION')+baseenv.subst('$EXTVERS'))
```

Keeping forks up-to-date



$ git remote add upstream git@github.com:JeffersonLab/hcana.git
$ git fetch upstream
$ git merge upstream/develop

Issue Tracking in Git



X

- Issues to be solved can be created at any time, and assigned to a particular developer
- When pull requests are made, can be associated with one or more issues, and issue can be closed (if appropriate)

A new build system - SCons

- Traditionally, Hall A/C software has been built with "make"
  - Platform/system/compiler dependent configuration handled within Makefiles (coupled with #ifdef statements within the code itself)
  - Dependency checking not included by default, and is based on timestamp.
  - Having an "autoconf"-like configuration is desirable, but GNU Autoconf is highly complex
  - Makefiles are platform-dependent, and incredibly cryptic – basically unreadable to non-experts – making changes and updates difficult
  - Libtool (management of libraries) not available for all platforms
- Is there something better out there?

A new build system - SCons

- SCons is an open-source software construction tool
  - Written entirely in Python – power of a real programming language in configuration and build scripts … plus, our students know and love Python!
  - Scripts are much more readable than Makefiles
  - Integrated functionality similar to Autoconf
  - Built-in support for C/C++, and easily extensible for other builders (ROOTCINT)
  - Built-in dependency-checking – based on MD5 signatures, and not timestamps – important for git.
  - Designed from the ground up for cross-platform builds
  - Currently used by the JLab DAQ group for EVIO

Major Projects using SCons

- ASCEND - A system modeling package for engineering
- Cantera – A toolkit for chemical kinetics and thermodynamics
- CLAM – A framework to develop sophisticated audio analysis
- FreeNOS – A microkernel operating system written in C++
- IntensityEngine – A platform for 3D games and virtual worlds
- Lumiera – A professional video editor
- Madagascar – Geophysical data processing
- Nsound – C++ audio synthesis framework
- openEHR – Electronic Health Record standard
- V8 – Google's open source Javascript engine
- YafaRay – An open source raytracing engine

SCons – Current status

- Build and configuration scripts have been written, tested, and committed for both PODD and HCANA
- Configuration checks for:
  - ROOT installation
  - gcc/g++ compiler installation and functionality
  - Platform-dependent compiler/linking flags (64/32 bit, Linux/MacOSX)
- Currently, we are maintaining the traditional Make system and SCons in parallel (for both PODD and HCANA)