



Open/Closed Software Development

Ben Morgan



Open Source vs Open Development

- Open Source
 - *Accessibility and licensing of the code itself.*
- Open Development
 - *Accessibility and management/governance of modifications to the code.*
- **An open source code might not be open to development**
 - *E.g. you can get the code, but the developer doesn't accept contributions*
- **A closed-source code might be open to development**
 - *FLUKA/MCNP are sort of in this category.*
 - *Here we take “development” to include requests for features, not just programming.*
- **“Open/Closed”** have positive/negative connotations but projects may have good reasons for choosing closed source and/or development.

Tools and Practices

- **Neither open/closed guarantees software quality!**
 - *An open source code might come with no instructions*
 - *A code being openly developed might not test contributions, or it might not prioritize features requested by the users.*
- Before even worrying about open/closed source development, **establishing basic development tools and practices is a necessity!**
 - *We've covering some of these this week.*
- Other than helping developers and contributors work effectively, these also assist in the **more challenging** tasks of **management and governance of who, what, and how of development.**
 - *The development process you establish becomes a foundation of this.*

Development challenges in TEPP software projects

- **Limited FTE**, often with high turnover from limited funding/short contracts
 - *“Get it working” pressure can lead to neglect of long term support items*
 - *Low bus factor: 1-2 key people leaving can completely stall a project*
- **Not Invented Here** syndrome
 - *Usually driven and exacerbated by the above*
 - *Fewer developers can mean less breadth of knowledge/experience*
- Often a high level of **overthinking/engineering**
 - ***Developers $\cong$ Users***
- Importance of software work to an experiment or the community **may not be recognized** by funding bodies (or even experiment members!)
 - *Can be particularly challenging for community wide projects*

Addressing Development Challenges

- No matter the size of the project, **implementing, maintaining and focussing on the foundational aspects is necessary:**
 - *Documentation (README/CONTRIBUTING, [Sphinx](#), [Doxygen](#))*
 - *Build and testing ([CMake](#)/[CTest](#)/etc)*
 - *Code style/quality tools ([clang-format](#)/[tidy](#), [Black](#)/[flake8](#))*
 - *Package management/deployment ([Conda](#)/[Pip](#)/[Spack](#)/[CVMFS](#) etc)*
 - *Version Control and Continuous Integration workflows*
- Might be obvious (*even boring*), but **neglecting these will build up high interest technical debt** that sooner or later will have to be **repaid in FTE**
 - *Been there, done/doing that*
- Recognition of the criticality of sustainable and reproducible software for research, **but never be afraid to repeat the case to funders!**

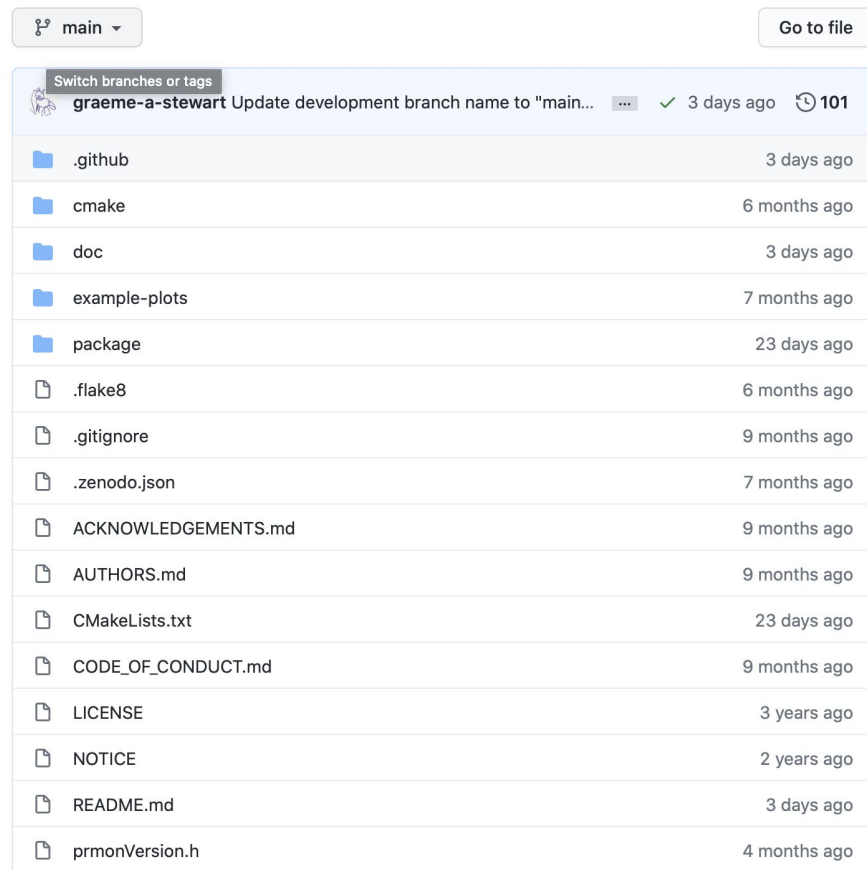
Tools and Suggestions for Effective Development

- These apply no matter the open-ness of the source or development
- ... and no matter whether there's 1 developer or 100...
- Caveats:
 - *Strong C++ bias*
 - *I'm just as guilty of not always doing what I suggest...*
 - *There will no doubt be a lot of "but..." and "what about..." questions*



Organizing the Project

- **Always** have needed “first contact” docs like README, CONTRIBUTING
- **Never** a need for a “special” layout of source or other code
 - *Languages often have a recommended/standard layout*
 - *Keep it simple, logical otherwise*
- Might seem trivial, but you want **principle of least surprise!**
 - *Minimize mental start up and context switching costs*



main

Switch branches or tags

graeme-a-stewart Update development branch name to "main... 3 days ago 101

.github	3 days ago
cmake	6 months ago
doc	3 days ago
example-plots	7 months ago
package	23 days ago
.flake8	6 months ago
.gitignore	9 months ago
.zenodo.json	7 months ago
ACKNOWLEDGEMENTS.md	9 months ago
AUTHORS.md	9 months ago
CMakeLists.txt	23 days ago
CODE_OF_CONDUCT.md	9 months ago
LICENSE	3 years ago
NOTICE	2 years ago
README.md	3 days ago
prmonVersion.h	4 months ago

Building the Project

- Always use a standard build tool
 - *CMake, Autotools, Setuptools*
- **Keep scripts simple and standard**
 - *Avoid bikeshedding and over-engineering!*
- **Build scripts are code** so document and maintain them as such
- **A simple, well maintained build system pays dividends in later items!**

 **amete** Update the release number to v2.2.0 ✓ History

 2 contributors

Raw Blame 📄 ✎ 🗑

100 lines (81 sloc) | 3.8 KB

```
1 # - Define the minimum CMake version
2 # HSF recommends 3.3 to support C/C++ compile features for C/C++11 across all
3 # platforms
4 cmake_minimum_required(VERSION 3.3)
5 # - Call project() to setup system
6 # From CMake 3, we can set the project version easily in one go
7 project(prmon VERSION 2.2.0)
8
9 #--- Define basic build settings -----
10 # - Use GNU-style hierarchy for installing build products
11 include(GNUInstallDirs)
12
13 # Add some build option variables, for static builds and profiling
14 if(NOT BUILD_STATIC)
15     set(BUILD_STATIC OFF)
16 endif()
17 set(BUILD_STATIC "${BUILD_STATIC}")
18 CACHE BOOL "Build a static version of the prmon binary" FORCE)
19
20 if(NOT PROFILE_GPROF)
21     set(PROFILE_GPROF OFF)
22 endif()
23 set(PROFILE_GPROF "${PROFILE_GPROF}")
24 CACHE BOOL "Build with the GNU profile option set" FORCE)
25
26 if(NOT PROFILE_GPERFTOOLS)
27     set(PROFILE_GPERFTOOLS OFF)
```


Testing the Project

- Always use a common/standard testing framework
 - E.g. [Catch2](#)/[googletest](#)
 - *stdout+Mk1 eyeball isn't!*
- **Test-driven development** can be very useful, both to
 - *Ensure tests are written(!)*
 - *Help clarify interfaces/contracts*
- Write tests to **triage+fix bugs**
- Ensure tests are **easily built/run as part of the development workflow**
 - E.g. *make test*

 **drbenmorgan** Bump Catch to 2.12.2 to support GCC9 ... X

History

1 contributor

Raw Blame

83 lines (68 sloc) 2.76 KB

```
1 #include "catch.hpp"
2
3 #include "falaise/quantity.h"
4
5 #include "bayeux/datatools/units.h"
6
7 TEST_CASE("Raw quantity construction", "") {
8     falaise::quantity x{};
9     REQUIRE(x() == 0.0);
10    REQUIRE_THROWS_AS(falaise::quantity(3.14, "furlong"), falaise::unknown_unit_error);
11 }
12
13 TEST_CASE("Raw quantity conversion", "") {
14     falaise::quantity x{3.14, "GeV"};
15     REQUIRE(x.dimension() == "energy");
16     REQUIRE(x.unit() == "GeV");
17     REQUIRE(x() == Approx(3.14 * CLHEP::GeV));
18     REQUIRE(x.value_in("eV") == Approx(3.14 * CLHEP::GeV / CLHEP::eV));
19     REQUIRE_THROWS_AS(x.value_in("km"), falaise::wrong_dimension_error);
20 }
21
22 TEST_CASE("Explicit dimensions construction", "") {
23     REQUIRE_THROWS_AS(falaise::mass_t(3.14, "km"), falaise::wrong_dimension_error);
24     REQUIRE_THROWS_AS(falaise::mass_t(3.14, "foo"), falaise::unknown_unit_error);
25
26     // Default construction has to have a sensible defaults!
```

Documenting the Project

- Includes README, CONTRIBUTING, INSTALL files etc.
- Use standard tools like [Doxygen](#), [Sphinx](#) for API/User Guides
- **Like tests, document-as-you-go**
 - *Will be clearer at time of writing*
 - *Helps you to clarify your design*
- **“Compile” docs as part of build**
 - *Check for mistakes!*
 - *Helps in release stage*
- Encourage users to contribute - **they bring a different perspective**

```
-- Show the previous page
20 #ifndef FALAISE_BOUNDED_INT_H
21 #define FALAISE_BOUNDED_INT_H
22
23 #include <stdint>
24 #include <exception>
25
26 namespace falaise {
27     ///! \brief Class representing an integer value with strict bounds
28     /*!
29     * \tparam lower_bound Inclusive lower bound on value
30     * \tparam upper_bound Inclusive upper bound on value
31     *
32     * Provides a convenience class for integral values with strict bounds that
33     * cannot overflow. The canonical use case in Falaise is indices for tracker
34     * layers and columns, [0,8] and [0,112] respectively. These bounds do not line
35     * up with the range of any builtin integral type, and builtin types are
36     * vulnerable to overflow/underflow. Instead of having to write code that checks
37     * bounds explicitly everytime we use an builtin integral :
38     *
39     * ```cpp
40     * const int32_t NLAYER = 8;
41     *
42     * void example(int32_t layerIndex) {
43     *     if (layer < 0 || layer > NLAYER) {
44     *         throw std::out_of_range;
45     *     }
46     *     ... do something ...
47     * }
48     * ```
49     *
50     * bounded_int allows us to do
51     *
52     * ```cpp
53     * using layer_index_t = bounded_int<0,8>;
54     *
55     * void example(layer_index_t i) {
56     *     ...
57     * }
```

Formatting the Code

- Enforce a coding style to ensure consistency and familiarity
 - E.g. [clang-format](#) for C/C++, [Flake8](#) for Python
- Use of a tool reduces the chances of a holy war over spaces/braces
 - *An area developers can be pointlessly opinionated about*
- Use [precommit](#) and IDE integrations to format on save/commit.

Falaise / .clang-format



drbenmorgan Bump column width to 100 ...

History

1 contributor

Raw

Blame



98 lines (97 sloc) | 2.78 KB

```
1  ---
2  Language:      Cpp
3  # BasedOnStyle: Google
4  AccessModifierOffset: -1
5  AlignAfterOpenBracket: Align
6  AlignConsecutiveAssignments: false
7  AlignConsecutiveDeclarations: false
8  AlignEscapedNewlinesLeft: true
9  AlignOperands: true
10 AlignTrailingComments: true
11 AllowAllParametersOfDeclarationOnNextLine: true
12 AllowShortBlocksOnASingleLine: false
13 AllowShortCaseLabelsOnASingleLine: false
14 AllowShortFunctionsOnASingleLine: All
15 AllowShortIfStatementsOnASingleLine: true
16 AllowShortLoopsOnASingleLine: true
17 AlwaysBreakAfterDefinitionReturnType: None
18 AlwaysBreakAfterReturnType: None
19 AlwaysBreakBeforeMultilineStrings: true
20 AlwaysBreakTemplateDeclarations: true
21 BinPackArguments: true
22 BinPackParameters: true
23 BraceWrapping:
```

Analysing the Code

- Use **static analysers** to suggest, or apply, fixes for style, performance
 - E.g. [clang-tidy](#) (C/C++)
- Usually easy to integrate with the build or test systems
 - *CMake makes clang-tidy use really easy, for example*
- Particularly useful for modernizing or simplifying older projects!

 drbenmorgan Initial clang-tidy config for Falaise ... ✓ History

🔍 1 contributor

Raw Blame



159 lines (158 sloc) 7.47 KB

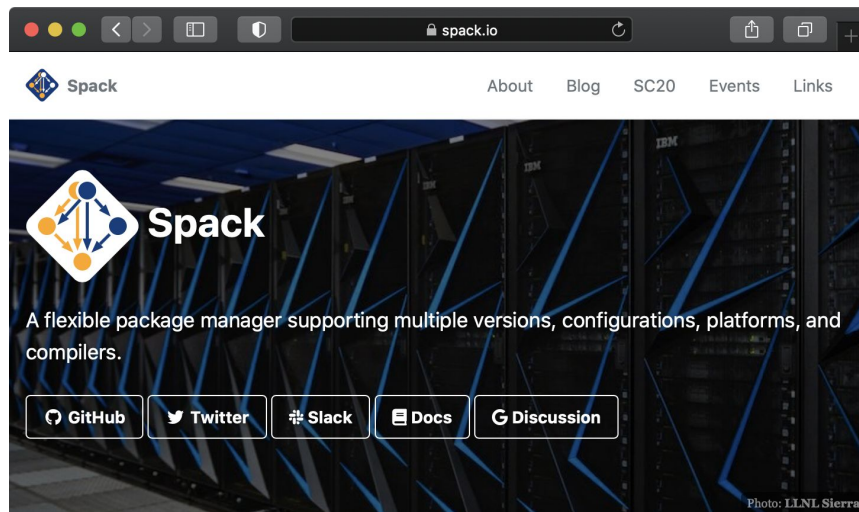
```
1 ---
2 Checks: 'clang-diagnostic-*,clang-analyzer-*,*,readability-*,modernize-*,perf
3 WarningsAsErrors: ''
4 HeaderFilterRegex: ''
5 AnalyzeTemporaryDtors: false
6 FormatStyle: file
7 CheckOptions:
8   - key: cert-dcl16-c.NewSuffixes
9     value: 'L;LL;LU;LLU'
10  - key: cert-oop54-cpp.WarnOnlyIfThisHasSuspiciousField
11    value: '0'
12  - key: cppcoreguidelines-explicit-virtual-functions.IgnoreDestructors
13    value: '1'
14  - key: cppcoreguidelines-non-private-member-variables-in-classes.IgnoreCl
15    value: '1'
16  - key: google-readability-braces-around-statements.ShortStatementLines
17    value: '1'
18  - key: google-readability-function-size.StatementThreshold
19    value: '800'
20  - key: google-readability-namespace-comments.ShortNamespaceLines
21    value: '10'
22  - key: google-readability-namespace-comments.SpacesBeforeComments
23    value: '2'
24  - key: modernize-loop-convert.MaxCopySize
```

Using Other Software in the Project

- All non-trivial projects use **external packages - dependencies**
 - *Already seen these with build/test/documentation ones!*
- **When the project needs an “X”, find a suitable off-the-shelf “X” first**
 - *“Suitable” means “Meets the requirements”*
 - *“I don’t like the implementation/style” **isn’t** a requirement*
 - ***One on the biggest time drains for projects can be the implementation, maintenance, and development of these new wheels***
 - *You can (and should!) contribute back to the projects you use!*
- Make sure the project is not locked to a specific version of the external
 - *All languages have constructs to compile/branch on the version*
- A minimum, or range, of supported versions is fine though
 - *Most build systems support this, .e.g CMake’s [find_package](#)*

Package Managers

- Always install dependencies with a widely used ***package manager***
 - *Don't write one yourself, please...*
- Choice will depend on languages used and platforms targeted
 - *Language specific, e.g. pip*
 - *Platform specific, e.g. apt*
 - *General(ish), e.g. conda, spack*
- Use a “requirements” file/package
 - *Lists packages and versions for a given “environment”*
 - *Aids easy setup/reproducibility*



Welcome to Spack!

Spack is a package manager for [supercomputers](#), Linux, and macOS. It makes installing scientific software easy. Spack isn't tied to a particular language; you can build a software stack in [Python](#) or R, link to libraries written in C, C++, or Fortran, and easily [swap compilers](#) or target [specific microarchitectures](#). Learn more [here](#).

Recent Posts

[Spack User Survey 2020](#)

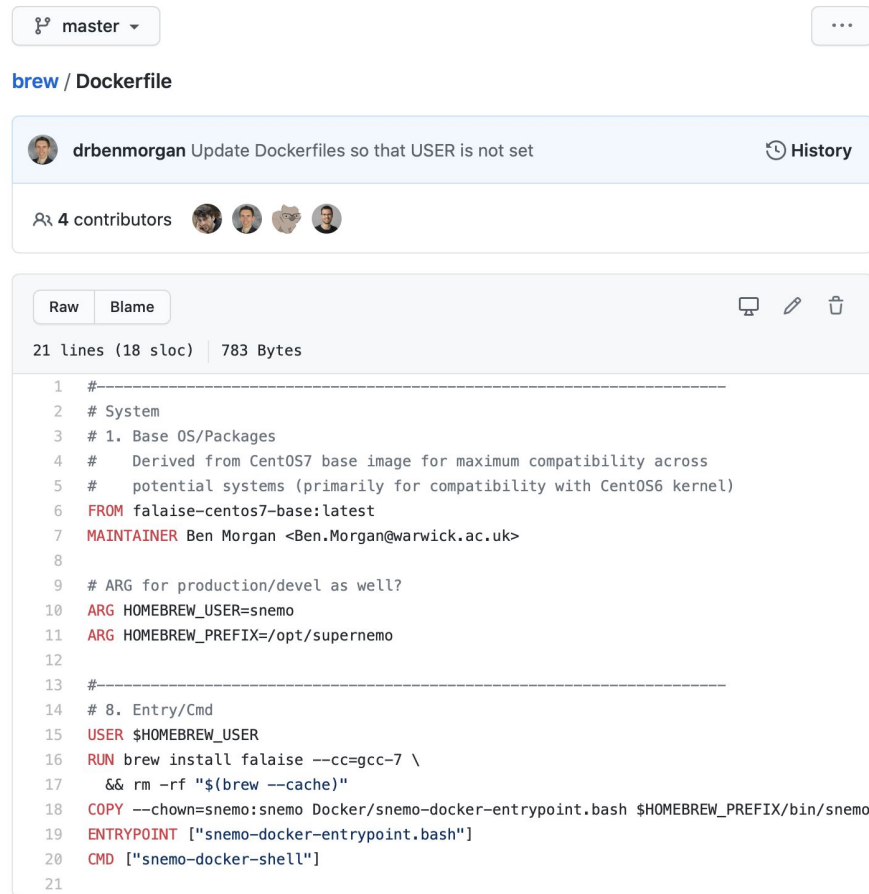
Results from the Spack 2020 User Survey are out! Check out our summary below.

[Spack featured on The Next Platform](#)

The Next Platform provides in-depth coverage of high-end computing at large enterprises, supercomputing centers, hyperscale data

Docker/Apptainer

- Containers can aid in distributing a development/use environment
 - *Modern IDEs can use them seamlessly (e.g. [VSCode](#))*
- Package manager “environment” files plus VCS/Container tags provide a great system for reproducible setups
- **Don’t** use containers as a sticking plaster over a bad build/packaging system - **fix these first!**



master ▾

brew / Dockerfile

drbenmorgan Update Dockerfiles so that USER is not set History

4 contributors

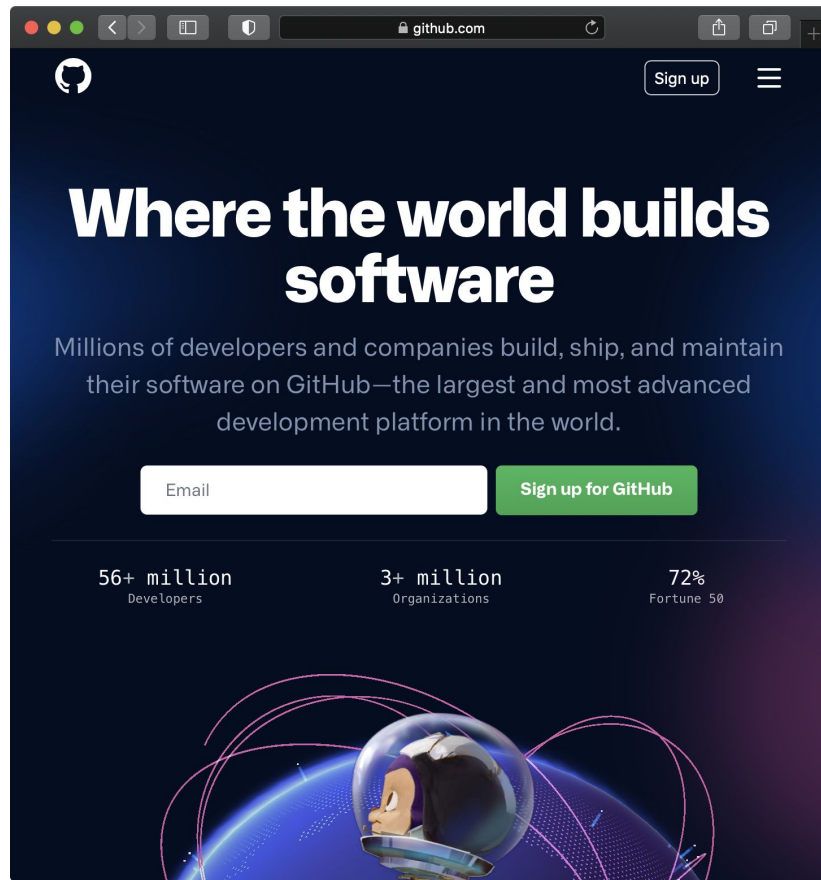
Raw Blame

21 lines (18 sloc) 783 Bytes

```
1 #-----
2 # System
3 # 1. Base OS/Packages
4 #   Derived from CentOS7 base image for maximum compatibility across
5 #   potential systems (primarily for compatibility with CentOS6 kernel)
6 FROM falaise-centos7-base:latest
7 MAINTAINER Ben Morgan <Ben.Morgan@warwick.ac.uk>
8
9 # ARG for production/devel as well?
10 ARG HOMEBREW_USER=snemo
11 ARG HOMEBREW_PREFIX=/opt/supernemo
12
13 #-----
14 # 8. Entry/Cmd
15 USER $HOMEBREW_USER
16 RUN brew install falaise --cc=gcc-7 \
17     && rm -rf "$(brew --cache)"
18 COPY --chown=snemo:snemo Docker/snemo-docker-entrypoint.bash $HOMEBREW_PREFIX/bin/snemo-
19 ENTRYPOINT ["snemo-docker-entrypoint.bash"]
20 CMD ["snemo-docker-shell"]
21
```

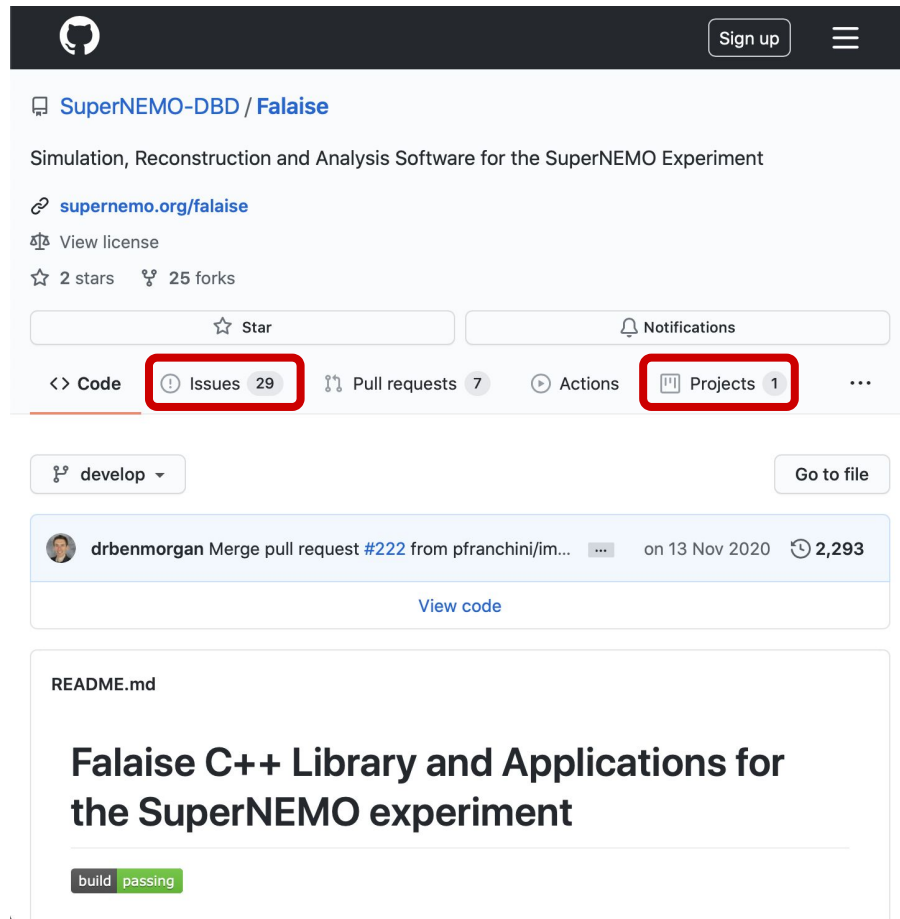

Hosting the Project

- **Git+GitHub** is the primary VCS and hosting solution, though many **GitLab instances at Institutes/Labs**
 - *These may be better if the project has specialist needs like access to GPU hardware for CI*
- Git repos are trivial to move, so you can always change later!
- **Never manage your own hosting** - you **cannot** do it as well as GitHub/Lab or Institute/Labs



Use Issues and Boards

- Use [Issues](#) as **primary** channel of communication to reduce “*where do I go to...*” and help engagement
 - *Email/Slack fine, **but encourage use of Issues!***
- [Discussions](#) feature may be better for “*How do I*” questions
- Use [Projects](#) to organise core tasks
 - *Can add Issues to specific Projects e.g. “Next Release”*
 - *Helps focus effort and reduce context switching cost*



github.com

HSF / phoenix

Unwatch 13 Star 12 Fork 16

Code Issues 19 Pull requests 2 Discussions Actions Projects Wiki Security Insights Settings

Add more realistic geometry for ATLAS #198

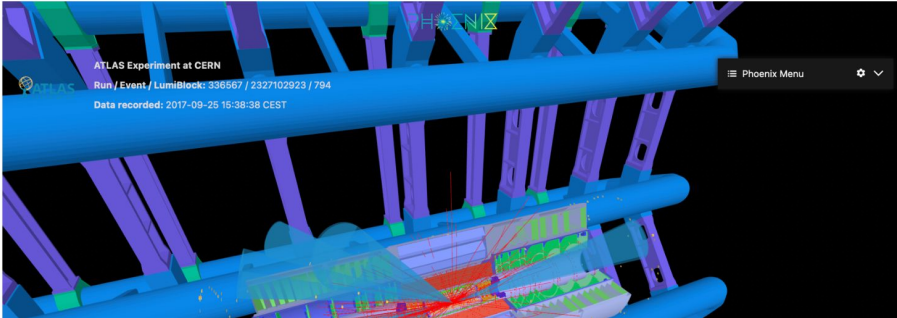
Draft EdwardMoyse wants to merge 11 commits into HSF:master from EdwardMoyse:new-geometry

Conversation 1 Commits 11 Checks 1 Files changed 36 +14,766 -28

EdwardMoyse commented on 17 Dec 2020 • edited Collaborator

This geometry was taken from a version of Tracer from June 2020 (it was not a direct copy, since back then the geometry was written in json format. Instead we converted this to glTF). Many thanks to GTU for providing the more detailed geometry!

This also contains some fixes for the hierarchical menus, and a new default configuration.



Reviewers

9inpachi

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

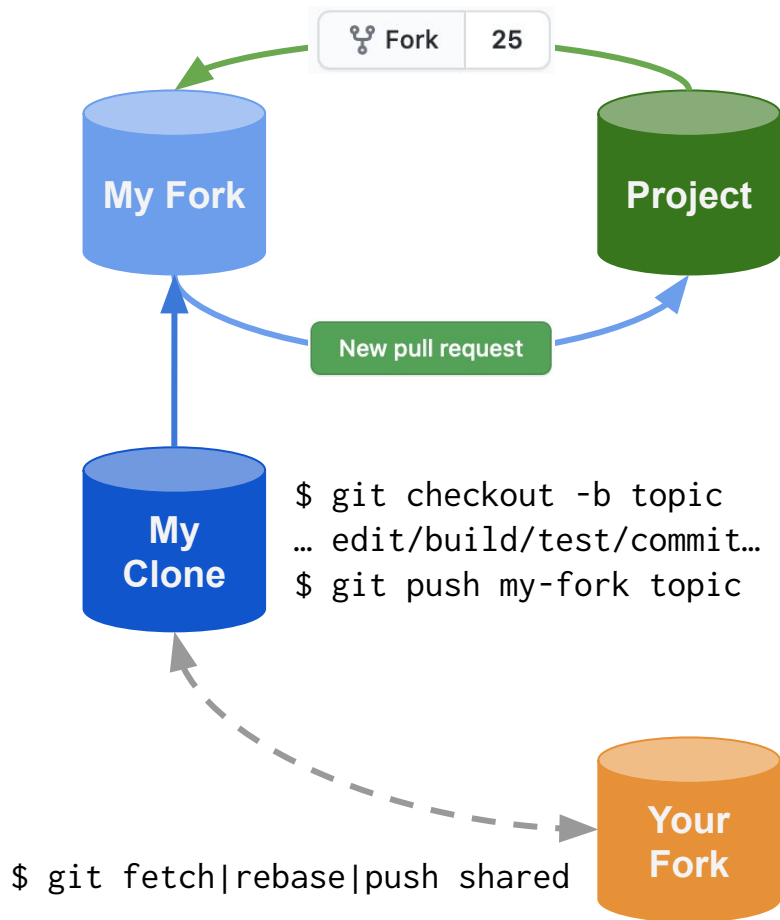
Successfully merging this pull request may close

Developing the Code: Topic Branches and Pull Requests

Never commit/push directly to the main branch - you will break it at some point!

Organising Topic Branches

- Forking Workflow simplest
 - *Topic Branches on Forks*
 - **Project repo only for merging**
- Tempting to push Topic Branches to project repo itself
 - *Fine for 1-2 developers ...*
 - *... but does not scale well*
 - *Forks **don't** stop co-development*
- **Even lead developers should submit work from Forks!**
 - *Help separate development work from release/maintenance tasks*



Testing Pull Requests

- Always build and test each PR using a CI system
 - E.g. [GitHub Actions](#), [Azure](#)
- A critical time saver, even if only targeting a single platform/OS
 - Will quickly reveal any coding oversights or “works on my machine” gremlins
- This is where earlier work on build and test systems starts to return the time investment!

[cmake-compile-features](#) / [.github](#) / [workflows](#) / [ci.yml](#)



drbenmorgan Improve organization of Ubuntu/macOS/Windows builds (#8) ...

History

1 contributor

Raw

Blame



79 lines (75 sloc) | 3.47 KB

```
1 name: CI
2
3 on: [pull_request]
4
5 jobs:
6   CentOS7-CVMFS:
7     # Run on latest GitHub Hosted Linux so
8     runs-on: [ubuntu-latest]
9     steps:
10      # Install and start CVMFS, possible place for action-ization ('setup-cvmfs')
11      - name: Install and Start CVMFS
12        run: |
13          sudo apt-get install lsb-release
14          wget https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest_all.
15          sudo dpkg -i cvmfs-release-latest_all.deb
16          rm -f cvmfs-release-latest_all.deb
17          sudo apt-get update
18          sudo apt-get install cvmfs cvmfs-config-default
19          sudo cvmfs_config setup
20          echo "CVMFS_REPOSITORIES=sft.cern.ch" | sudo tee -a /etc/cvmfs/default.local
21          echo "CVMFS_HTTP_PROXY=DIRECT" | sudo tee -a /etc/cvmfs/default.local > /d
22          sudo service autofs restart
23          sudo cvmfs_config probe
24      # Like Azure, if we have to run steps mixed between VM and Container, Container m
```

Now a much simpler way!

[cvmfs-contrib/github-action-cvmfs/](#)

The screenshot shows a GitHub Pull Request (PR) for the repository 'drbenmorgan / cmake-compile-features'. The PR title is 'Improve organization of Ubuntu/macOS/Windows builds #8'. It is marked as 'Merged' and shows that 9 commits were merged into the 'master' branch 5 days ago. The PR has 1 conversation, 9 commits, 10 checks, and 2 files changed. A status bar at the top right shows '+31 -66' with a red progress indicator.

Below the PR details, the CI status is shown as 'on: pull_request' with a green circle icon. A list of CI jobs is displayed, with 'CentOS7-CVMFS' selected. The job status is 'failed 5 days ago in 1m 48s'. The job log shows the following output:

```
101 -- Checking support for "cxx_random" in C++17: works
102 -- Checking support for "cxx_thread_thread" in C++17
103 -- Checking support for "cxx_thread_thread" in C++17: works
104 CMake Error at CMakeLists.txt:122 (target_compile_features):
105   target_compile_features The compiler feature "cxx_std_17" is not known to
106   CXX compiler
107
108   "GNU"
109
110   version 4.8.5.
111
112
113 -- Performing Test COMPILER_HAS_HIDDEN_VISIBILITY
```

GitHub Actions in Action

Real value is the [record of build/test results](#), so PR author can find/fix issues quickly

CI Scripting

- Like previous recommendations, don't overthink/engineer these
 - *Each system does have slight differences in functionality*
- Even complex builds don't require complex scripting
- They are part of the project code, so develop and maintain them with the same care
 - *Including submitting changes through PRs - the CI can test CI!*

```
46 UbuntuMacWindows:
47   strategy:
48     fail-fast: false
49     matrix:
50       # As the matrix is sparse, write explicitly rather than x-product and excludes
51       # Limit compilers AFAP to packages installable directly (no ppa)
52       include:
53         - { os: ubuntu-18.04, cc: gcc-7, cxx: g++-7 }
54         - { os: ubuntu-18.04, cc: gcc-8, cxx: g++-8 }
55         - { os: ubuntu-18.04, cc: clang-8, cxx: clang++-8 }
56         - { os: ubuntu-20.04, cc: clang-9, cxx: clang++-9 }
57         - { os: ubuntu-20.04, cc: gcc-9, cxx: g++-9 }
58         - { os: ubuntu-20.04, cc: gcc-10, cxx: g++-10 }
59         - { os: ubuntu-20.04, cc: clang-10, cxx: clang++-10 }
60         - { os: macos-10.15 }
61         - { os: windows-latest }
62     runs-on: ${ matrix.os }
63   steps:
64     - uses: actions/checkout@v2
65     - name: Setup build environment
66       # Set CC/CXX only if they exist
67       # See https://github.community/t/possible-to-use-conditional-in-the-env-section
68       # for why this can be done in env: above
69       shell: bash
70       run: |
71         [[ ! -z "${ matrix.cc }" ]] && echo "CC=${ matrix.cc }" >> $GITHUB_ENV ||
72         [[ ! -z "${ matrix.cxx }" ]] && echo "CXX=${ matrix.cxx }" >> $GITHUB_ENV
73     - name: Configure
74       run: |
75         cmake -S. -B ./build
76     - name: Build
77       run: |
78         cmake --build ./build --config RelWithDebInfo
```

Reviewing Pull Requests

- Use [PR reviews](#) as “peer review for code”
 - *Discuss the implementation, suggest revisions, improvements*
 - *Ask for addition of missing items like tests, documentation!*
- Can be light touch “fine for me”, to major revisions, even rejection
- Collaborative approach is valuable in increasing the Bus Factor and reducing over-engineering

Merged

Implement a Dead cells service #222

drbenmorgan merged 7 commits into [SuperNEMO-DBD:develop](#) from [pfranchini:implement-deadcells-service](#)



drbenmorgan on 6 Nov 2020

Member

...

Suggested change ⓘ

```
25 - enum status {
26 -     good = 0,           // good working cell
27 -     dead = 1,           // dead cells with dead anode
28 -     cathode_ground = 2, // cathode-to-ground short
29 -     cathode_cathode = 3, // cathode-to-cathode short
30 -     other = 4           // other
31 - };

25 + enum class cell_status {
26 +     good = 0,           // good working cell
27 +     dead = 1,           // dead cells with dead anode
28 +     cathode_ground = 2, // cathode-to-ground short
29 +     cathode_cathode = 3, // cathode-to-cathode short
30 +     other = 4           // other
31 + };
```

Should be an `enum class` to give stronger typing and clarity.



1



pfranchini on 6 Nov 2020

Author

Contributor

...

Ah @emchauve , we crossed over in our timings there! I think where we have a cathode-to-cathode short, we should identify whether it's at the top or bottom (we have this info already) and then just not activate those particular cathodes. But the rest of the cell should be fine.



Review required

[Add your review](#)

At least 1 approving review is required by reviewers with write access. [Learn more.](#)



Some checks were not successful

[Hide all checks](#)

2 in progress, 14 successful, and 1 failing checks



style and docs / documentation (pull_request) Successful in 5m

Required

[Details](#)



linux tests / rhel8-platform-python (pull_request) Successful in 10m

[Details](#)



linux tests / clingo (pull_request) Successful in 29m

Required

[Details](#)



linux tests / clingo-cffi (pull_request) Successful in 24m

[Details](#)



codecov/project — 80.15% (-2.44%) compared to 62f8087

[Details](#)



codecov/patch — Coverage not affected when comparing 62f8087...ebbef03

[Details](#)



Merging is blocked

Merging can be performed automatically with 1 approving review.

Enable auto-merge (squash)



Automatically merge when all requirements are met. [Learn more](#)

Merging Pull Requests

Use Actions and Reviews to build a “pre merge” checklist

Release & Deployment

- By testing/reviewing PRs before merging to the main branch, **each PR merge provides a potential new release for the project**
 - *When/how often to make a release, and whether detailed testing (e.g. physics validation) is required before tagging is highly project dependent*
- **For each merge to main and new tag (release) of the project, a CI task should be triggered to:**
 - *Create and publish the documentation for the project*
 - *Create package(s)/environment for the project, e.g. conda, spack, Docker*
- Last item invaluable if you need to run large validation jobs for releases!
- Might require extra resources (e.g. ReadTheDocs, DockerHub), **but earlier investment in standard tools simplifies integration!**
 - *And GitHub/Lab can often handle this for you (gh-pages, Packages)*

Simulation, Reconstruction and Analysis Software for the SuperNEMO Experiment

supernemo.org/falaise

View license

2 stars 25 forks

Star

Notifications

<> Code ! Issues 29 Pull requests 7 Actions Projects 1 ...

develop

Go to file

drbenmorgan Merge pull request #222 from pfranchini/im... on 13 Nov 2020 2,293

View code

README.md

Falaise C++ Library and Applications for the SuperNEMO experiment

build passing

Simulation, Reconstruction and Analysis Software for the SuperNEMO Experiment

supernemo.org/falaise

View license

2 stars 25 forks

Star

Notifications

<> Code ! Issues 29 Pull requests 7 Actions Projects 1 ...

gh-pages

Go to file

Falaise 4.0.1

SuperNEMO Software Toolkit

Main Page Related Pages Modules Namespaces Classes Files

Search

Welcome to the Falaise Documentation

Falaise is the software system for the SuperNEMO experiment. These pages document the usage of the core applications, together with guides on extending their functionality and the full C++ API of the Falaise extension system.

Getting Started

If you're reading this online and don't yet have an install of Falaise, a [Quickstart Guide](#) is available. Note that at present Falaise is only supported on POSIX platforms, and even here not all variants are guaranteed to work!

User Guides for Core Applications and Libraries

Once you have an install of Falaise, the following pages provide introductions to using and extending the core applications:

- [FLSimulate](#): guide for using the Geant4 based simulations of the SuperNEMO Demonstrator and Commissioning detector setups.
- [FLReconstruct](#) guide for using the pipeline application for reading, reconstructing and outputting data generated by the FLSimulate detector simulation.
 - [Data Structures output by FLReconstruct](#)
 - [Customizing the pipeline for FLReconstruct](#)
 - [Writing plugin modules for FLReconstruct](#)
 - [Working with RunEvent Data in plugin modules](#)
 - [Using Services to access metadata](#)
- [FLVisualize](#) guide for viewing simulated and reconstructed data.

Contributing to Falaise Development

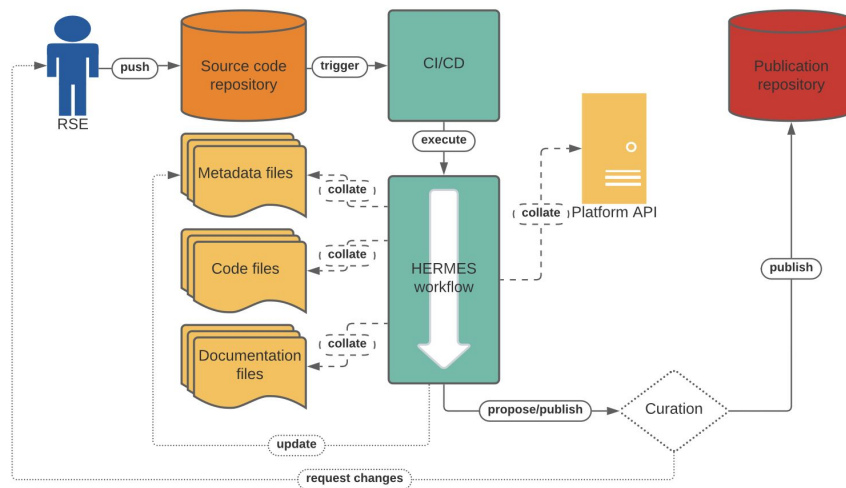
Contributions to the core of Falaise are welcome. Please visit our [GitHub repo](#) for further information.

Publishing Documentation

An easy way to do this is through [GitHub Pages](#)

Publishing Code

- We can publish releases on GitHub/Lab, but as we've discussed, will these platforms always be here?
- Can also publish a release to Zenodo for better longevity.
 - *And as a citable reference*
- This can also be automated in CI with tools like [HERMES](#)
 - *Richer metadata as well.*



geant4, geant4-data: versions 10.6.3 and 10.7.0 #20250

Edit

Open with ▾



Merged

sethrj merged 2 commits into `spack:develop` from `drbenmorgan:bump-geant4` on 4 Dec 2020

Conversation 5



Commits 2



Checks 13



Files changed 8

+38 -4



drbenmorgan commented on 4 Dec 2020

Member



Update geant4-data and individual datasets for Geant4 versions 10.6.3 and 10.7.0.

Update geant4 package with new versions 10.6.3 and 10.7.0. Update dependencies on CLHEP and VecGeom with versions required for Geant4 10.7.

Add `GEANT4_INSTALL_PACKAGE_CACHE=OFF` to CMake args for 10.6 onwards. Prevents install of the "package cache" file that contains hard-coded paths for dependencies, improving relocatability. It relies on Spack setting `CMAKE_PREFIX_PATH` correctly in build/use environments that consume the geant4 package. This has been tested locally in a simple environment to build a Geant4 example, and appears to function correctly.

Reviewers requested, also pinging @vvolkl and @ChristianTackeGSI for their info.



2



geant4, geant4-data: versions 10.6.3 and 10.7.0

✖ 62fc701

Reviewers



sethrj



ChristianTackeGSI



gartung



Assignees



No one—assign yourself

Labels



hep

new-version

Projects



None yet

Example Packaging: Spack

Especially valuable for community wide projects to have a presence in mainline package managers

Effective Collaborative Development

- Here we move from the easy technical side to the harder human side.
 - *How do collaborate effectively?*
 - *How do we ensure the longevity of the project?*
 - *How do we make decisions?*



How do we *actually* develop code?

- Our own projects that only we use
 - *We are basically self-governing...*
- A project we share with a few people in our own institute
 - *We make decisions over coffee/chats in the corridor*
- ...
- A project developed/used in multiple institutes across multiple countries
 - *Who can contribute to the code and how?*
 - ***“Contributors”***
 - *Who can make changes to the code, that is “merge to main”, or “make a release” (i.e. decision/direction) making authority*
 - ***“Maintainers”***

How Governance Models arise in software projects

- “Governance” is roughly the model we use for the conventions and rules that determine who can do what, how and when to the software project.
 - *Projects generally start small so certain governance models arise naturally/evolutionarily*

Do-Ocracy

Decisions by those who
“do the most”.

Single Vendor

Company/Lab/University

Foundation

Take input from multiple
vendors/stakeholders to
drive decisions.
Often non-profit.

Founder-Leader

Original author(s) make
all the decisions

Council/Board

Decision by committee.
Might be elected or
appointed

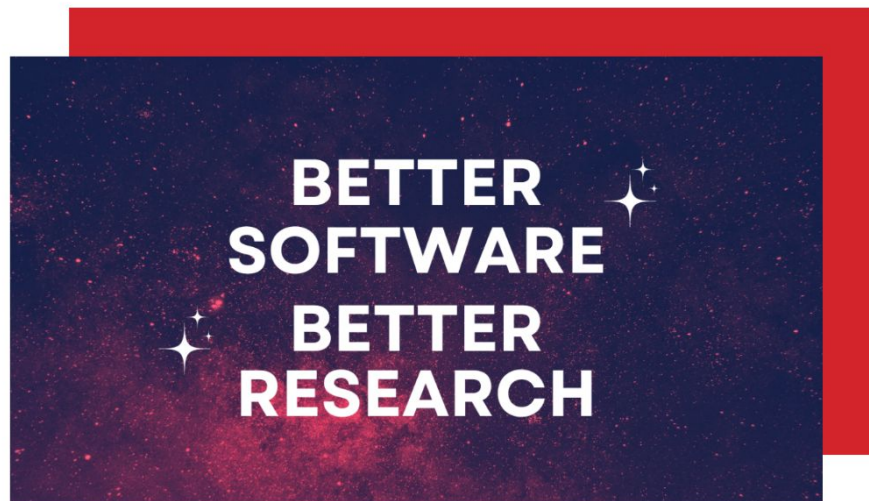
Does this really matter for TEPP?

- If your project is small, it's still useful to at least have
 - *Clear README.md, CONTRIBUTING.md, and CODE_OF_CONDUCT.md so the process of development, and who makes decisions, is clear.*
 - *Use Git and the described development workflow to organise this process so changes are clear, tested, and recorded (what was done and why).*
 - *Even if it's just you and a colleague, it avoids your work breaking theirs, and vice-versa.*
- As projects grow, whether closed or open, there usually comes a point when the more **structured Board/Vendor approach becomes helpful**.
 - *E.g. “Working Groups” in experiments like ATLAS*
 - *As much a **risk mitigation** strategy as governance: ensure effort is directed to meet scientific (user) needs, effort and expertise is distributed.*

About us

The Software Sustainability Institute is the first organisation in the world that was dedicated to improving software in research. We help people build better and more sustainable software to enable world-class research.

Read more >



How do I implement governance?

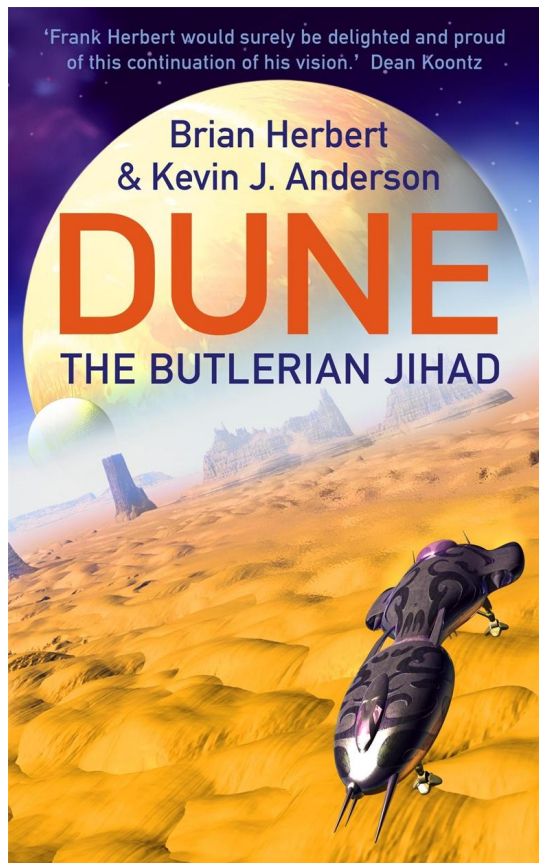
Software Sustainability Institute has several guides, and even some funding schemes.

The Open/Closed Paradigm in Scientific Software

- Plenty of closed source/development codes used day-to-day in research
 - *E.g. DAQ/laboratory codes: very specific, have little/no use outside one place.*
 - *That's why the development process is more important than open/closed*
- Open development **does not solve** sustainability problems like lack of funded effort, but it **can reduce barriers** to finding new contributors (and new sources of funding!).
 - *Naturally, the classic “more eyes make bugs shallow” also applies.*
- Open development **may** help to reduce wheel reinvention/silo-ization
 - *People using your code are encouraged to contribute back.*
- **There is one important elephant in the matrix...**

Thou shalt not make a machine in the likeness of a human mind

- “AI slop” PRs are a thing now - and is beginning to impact open development
 - *Overloading existing maintainers*
 - *Quality can be very low*
- Even started appearing in some scientific code - small now, but will only grow.
- Impacts your own projects and those you contribute too
 - *You might have to deal with this.*
 - *Contributions may have stricter rules.*
- **No easy answer or solution at this stage...**



In Summary

- **Whether your code is open/closed invest time in simple and standard tools** and methods to reduce developer overhead and points of failure, and increase automation
- **Use GitHub/Lab's tools** to provide a simple yet highly capable platform for open or closed development that returns investment in standard tools
- **Several models for Governing open/closed development:** ultimately a mutually agreed set of rules and procedures that outline who is responsible for what in the code, and who can integrate changes.

