

Organizing Small Software Projects



Ben Morgan

What is a "Small Software Project"?

- In the context of this meeting, software projects that may be
 - **Specific**: e.g. to simulate/process/analyse data from an experiment
 - *General*: e.g. event generators used by many experiments
- The metric here for "Small" is **total developer FTE**
 - Not always related to size of the experiment/community using the software
 - FTE because developers may **only be funded a fraction of their time** to work on the software/experiment
 - This is slightly biased towards specific projects, as general ones often have an active, though transient, population of contributors
- To take the SuperNEMO $0\nu\beta\beta$ search experiment as an example:
 - ~100 researchers, 1-2FTE on software from 2-8 developers

Typical challenges faced in smaller 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*
 - Easily leads to a critical Bus Factor of 1-2
- **Not Invented Here** syndrome (*reinventing the wheel*)
 - Usually driven and exacerbated by the above
 - Fewer developers can mean less breadth of knowledge/experience
- Often a high level of **overthinking/engineering**
 - Not unique to small projects, but creates further long term support issues
 - Developers ≅ 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 Challenges

- In all cases, the challenges are best addressed by projects implementing, maintaining and focussing on the foundations
 - Documentation (README/CONTRIBUTING, <u>Sphinx</u>, <u>Doxygen</u>)
 - Build and testing (<u>CMake/CTest</u>/etc)
 - Code style/quality tools (<u>clang-format/tidy</u>, <u>Black/flake8</u>)
 - Package management/deployment (<u>Conda/Pip/Spack/CVMFS</u> 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
- There is a growing recognition of the criticality of sustainable and reproducible software for research, **but never be afraid to repeat the case!**

Tools and Suggestions for Small Projects

- Managing Code and Documents
- Development Workflows
- Release and Deployment
- 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 (please write them in the Live Notes, they are **always** welcome!)



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

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
Dackage	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
	3 years ago
NOTICE	2 years ago
B README.md	3 days ago
P prmonVersion.h	4 months add

Go to file

₽ main ▼

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!



Testing the Project

- Always use a common/standard testing framework
 - E.g. <u>Catch2/googletest</u>
 - 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 — × ③ History

A: 1 contributor

Raw

Blame

P: 1 lines

P: 1 lines

P: 1 lines
```

```
83 lines (68 sloc) 2.76 KB
     #include "catch.hpp"
     #include "falaise/quantity.h"
     #include "bayeux/datatools/units.h"
     TEST CASE("Raw quantity construction", "") {
       falaise::guantity x{};
       REOUIRE(x() == 0.0);
       REQUIRE THROWS AS(falaise::guantity(3.14, "furlong"), falaise::unknown unit error);
     TEST CASE("Raw quantity conversion", "") {
       falaise::quantity x{3.14, "GeV"};
       REOUIRE(x.dimension() == "energy");
       REQUIRE(x.unit() == "GeV");
       REOUIRE(x() == Approx(3.14 * CLHEP::GeV));
       REQUIRE(x.value in("eV") == Approx(3.14 * CLHEP::GeV / CLHEP::eV));
       REOUIRE THROWS AS(x,value in("km"), falaise::wrong dimension error);
20
    3
     TEST_CASE("Explicit dimensions construction", "") {
       REOUIRE THROWS AS(falaise::mass t(3.14, "km"), falaise::wrong dimension error);
       REQUIRE_THROWS_AS(falaise::mass_t(3.14, "foo"), falaise::unknown_unit_error);
       // Default construction has to have a sensible defaults!
       feleiser, length + 1().
```

Documenting the Project

- Includes README, CONTRIBUTING, INSTALL files etc.
- Use standard tools like <u>Doxygen</u>, <u>Sphinx</u> 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
     #ITHOET FALAISE BUUNDED INT H
     #define FALAISE_BOUNDED_INT_H
     #include <cstdint>
    #include <exception>
     namespace falaise {
     //! \brief Class representing an integer value with strict bounds
28
     /*
      * \tparam lower bound Inclusive lower bound on value
      * \tparam upper_bound Inclusive upper bound on value
      *
      * Provides a convenience class for integral values with strict bounds that
      * cannot overflow. The canonical use case in Falaise is indices for tracker
      * lavers and columns. [0.8] and [0.112] respectively. These bounds do not line
34
      * up with the range of any builtin integral type, and builtin types are
      * vulnerable to overflow/underflow. Instead of having to write code that checks
      * bounds explicitly everytime we use an builtin integral :
38
      *
      * ```cpp
      * const int32 t NLAYER = 8;
      * void example(int32 t laverIndex) {
         if (layer < 0 || layer > NLAYER) {
            throw std::out of range;
         ι
         ... do something ...
      * }
      * bounded int allows us to do
     * ```cpp
      * using layer_index_t = bounded_int<0,8>;
54
      * void example(layer_index_t i) {
```

Formatting the Code

- Enforce a coding style to ensure consistency and familiarity
 - E.g. <u>clang-format</u> for C/C++, <u>Flake8</u> 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 **Git hooks and IDE integrations** to format on save/commit

Falaise / .clang-format



Analysing the Code

• Use **static analysers** to suggest, or apply, fixes for style, performance

• E.g. <u>clang-tidy</u> (C/C++)

- Usually easy to integrate with the build or test systems
 - CMake makes clang-tidy use really easy, <u>for example</u>
- Particularly useful for modernizing or simplifying older projects!

Falaise / .clang-tidy

9	drbenmorgan Initial clang-tidy config for Falaise						
୧ኣ 1	R 1 contributor						
Rav	w Blame	· · · · · · · · · · · · · · · · · · ·					
159	lines (158 sloc)	7.47 КВ					
1							
2	Checks:	'clang-diagnostic-*,clang-analyzer-*,-*,readability-*,modernize-*,perf					
3	WarningsAsError	s: ''					
4	HeaderFilterReg	ex: ''					
5	AnalyzeTemporar	yDtors: 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	- кеу:	google-readabllity-braces-around-statements.shortStatementLines					
1/	value:	1. Tanala madakilitu function sina CtatamantThrashald					
18	- кеу:	google-readablilty-tunction-size.StatementIhreshold					
19	value:	ovv					
20	- Key:	yoogie-reauabitity-namespace-comments.shortwamespaceLines					
21	- kov	10 google_readability_namesnace_comments_SnacesRefereComments					
22	- Key:	goog te-readabitity-namespace-comments, spacesberorecomments					
23	value:	2 madarniza laan convert MayConvCiza					
24	- кеу:	mouernize-toop-convert.maxtopySize					

Using Other Software in the Project

- All 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 a small project 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 <u>find_package</u>

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 <u>supercomputers</u>, Linux, and macOS. It makes installing scientific software easy. Spack isn't tied to a particular language; you can build a software stack in <u>Python</u> or R, link to libraries written in C, C++, or Fortran, and easily <u>swap compilers</u> or target <u>specific microarchitectures</u>. Learn more <u>here</u>.

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/Singularity

- Containers can aid in distributing a development/use environment
 - Modern IDEs can use them seamlessly (e.g. <u>VSCode</u>)
- Package manager "environment" files and VCS/Container tags provide great system for reproducible setups
- Don't use containers as a sticking plaster over a bad build/packaging system - fix these first!

brew / Dockerfile drbenmorgan Update Dockerfiles so that USER is not set (1) History 83 4 contributors Blame Raw 21 lines (18 sloc) 783 Bytes System # 1. Base OS/Packages Derived from CentOS7 base image for maximum compatibility across potential systems (primarily for compatibility with CentOS6 kernel) FROM falaise-centos7-base:latest MAINTAINER Ben Morgan <Ben.Morgan@warwick.ac.uk> # ARG for production/devel as well? ARG HOMEBREW_USER=snemo ARG HOMEBREW PREFIX=/opt/supernemo # 8. Entry/Cmd USER \$HOMEBREW USER RUN brew install falaise --cc=qcc-7 \ && rm -rf "\$(brew --cache)" 18 COPY --chown=snemo:snemo Docker/snemo-docker-entrypoint.bash \$HOMEBREW_PREFIX/bin/snemo ENTRYPOINT ["snemo-docker-entrypoint.bash"] 20 CMD ["snemo-docker-shell"]

₽ master

Hosting the Project

- <u>Git</u>+<u>GitHub</u> 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

Where the world builds software

aithub.com

Sian up

 \Box

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.



Use Issues and Boards

- Use <u>Issues</u> as primary channel of communication to reduce "where do I go to..." and help engagement
 - Email/Slack fine, but encourage use of Issues!
 - Beta "<u>Discussions</u>" feature may be good for "How do I" questions
- Use **<u>Projects</u>** to organise core tasks
 - Can add Issues to specific
 Projects e.g. "Next Release"
 - Helps focus effort and reduce context switching cost

SuperNEMO-DBD / Falaise

Simulation, Reconstruction and Analysis Software for the SuperNEMO Experiment

ି superne	emo.org/falaise					
∑ View lice	ense					
☆ 2 stars	약 25 forks					
	☆ Star			Ĺ	Notifications	
<> Code	! Issues 29	ູ່ໃງ Pull requests	7	Actions	Projects 1	
ド devel	op -					Go to file
🌍 drbe	enmorgan Merge pull re	equest #222 from p	franch	nini/im	on 13 Nov 2020	3,293
View code						
README	.md					
Falaise C++ Library and Applications for the SuperNEMO experiment						
build	passing					
					. /	



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 and best
 - Topic Branches on Forks
 - Project repo only for merging
- Tempting to push Topic Branches to project repo itself
 - Fine for 1-3 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
ନ୍ୟ 1 (contributor
Rav 79 li	/ Blame ines (75 sloc) 3.47 KB
1	name: CI
2	
3	on: [pull_request]
4	Now a much simpler way!
6	CentOS7_CVMES cvmfs-contrib/github-action-cvm
7	# Run on latest GitHub Hosted Linux so fs /
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	<pre>sudo apt-get install lsb-release</pre>
14	<pre>wget https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest_all.</pre>
15	<pre>sudo dpkg -i cvmfs-release-latest_all.deb</pre>
16	<pre>rm -f cvmfs-release-latest_all.deb</pre>
17	sudo apt-get update
18	<pre>sudo apt-get install cvmfs cvmfs-config-default</pre>
19	<pre>sudo cvmfs_config setup</pre>
20	<pre>echo "CVMFS_REPOSITORIES=sft.cern.ch" sudo tee -a /etc/cvmfs/default.loca</pre>
21	<pre>echo "CVMFS_HTTP_PROXY=DIRECT" sudo tee -a /etc/cvmfs/default.local > /c</pre>
22	sudo service autofs restart
23	<pre>sudo cvmfs_config probe</pre>
24	# Like Azure, if we have to run steps mixed between VM and Container, Container m
	The second se

	🗊 🔒 github.com	
📮 drbenmorgan / cmake-com	pile-features	ⓒ Unwatch ▾ 3 ਪੋ ਨੇ Star 7 ²⁹ Fork 1
<> Code 13 Pull requests (>)	Actions 🕕 Security 🗠 Insights 🕸 Settings	
Improve organization	n of Ubuntu/macOS/Windows bui	Lilds #8
□ Conversation □ ··· Commits	E Checks 10 E Files changed 2	+31-66
O Use bash explicitly for setting up env	5024f80 🕶	🙄 Re-run jobs 👻
✓ CI [on: pull_request	2 CentOS7-CVMFS failed 5 days ago in 1m 48s	Q Search logs ····
× CentOS7-CVMFS	✓ Suild CentOS/gcc4.8	285
✓ UbuntuMacWindows (ubuntu-1	101 — Checking support for "cxx_random" in C++17: 102 — Checking support for "cxx thread thread" ir	7: works in C++17
✓ UbuntuMacWindows (ubuntu-1	103 — Checking support for "cxx_thread_thread" ir 104 (Make Error at (Makelists.txt:122 (target com	in C++17: works mole features):
✓ UbuntuMacWindows (ubuntu-1	105 target_compile_features The compiler feature	re "cxx_std_17" is not known to
✓ UbuntuMacWindows (ubuntu-2		
✓ UbuntuMacWindows (ubuntu-2	108 "GNU" 109	
✓ UbuntuMacWindows (ubuntu-2	110 version 4.8.5. 111	
✓ UbuntuMacWindows (ubuntu-2	112 113 — Performing Test COMPILER_HAS_HIDDEN_VISIBIL	ILITY

GitHub Actions in Action

Real value is the <u>record of</u> <u>build/test results</u>, 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!

40	
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	<pre># Limit compilers AFAP to packages installable directly (no ppa)</pre>
52	include:
53	<pre>- { os: ubuntu-18.04, cc: gcc-7, cxx: g++-7 }</pre>
54	<pre>- { os: ubuntu-18.04, cc: gcc-8, cxx: g++-8 }</pre>
55	<pre>- { os: ubuntu-18.04, cc: clang-8, cxx: clang++-8 }</pre>
56	<pre>- { os: ubuntu-20.04, cc: clang-9, cxx: clang++-9 }</pre>
57	<pre>- { os: ubuntu-20.04, cc: gcc-9, cxx: g++-9 }</pre>
58	<pre>- { os: ubuntu-20.04, cc: gcc-10, cxx: g++-10 }</pre>
59	- { os: ubuntu-20.04, cc: clang-10, cxx: clang++-10 }
60	- { os: macos-10.15 }
61	<pre>- { os: windows-latest }</pre>
62	<pre>runs-on: \${{ matrix.os }}</pre>
63	steps:
64	- uses: actions/checkout@v2
65	– name: Setup build environment
66	<pre># Set CC/CXX only if they exist</pre>
67	<pre># See https://github.community/t/possible-to-use-conditional-in-the-env-section</pre>
68	<pre># for why this can be done in env: above</pre>
69	shell: bash
70	run:
71	<pre>[[! -z "\${{ matrix.cc }}"]] && echo "CC=\${{ matrix.cc }}" >> \$GITHUB_ENV </pre>
72	<pre>[[! -z "\${{ matrix.cxx }}"]] && echo "CXX=\${{ matrix.cxx }}" >> \$GITHUB_ENV</pre>
73	- name: Configure
74	run:
75	cmake -SB ./build
76	– name: Build
77	run:
78	<pre>cmakebuild ./buildconfig RelWithDebInfo</pre>
-	

Reviewing Pull Requests

- Use <u>PR reviews</u> 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



Implement a Dead cells service #222 drbenmorgan merged 7 commits into SuperNEMO-DBD:develop from pfranchini:implement-deadcells-servic

arbern	norga	an on 6 Nov 2020 Membe	Jer
Sugge	sted c	hange (i)	
25	-	enum status {	
26	-	good = 0,	// good working cell
27	-	dead = 1 ,	<pre>// dead cells with dead anode</pre>
28	-	<pre>cathode_ground = 2,</pre>	// cathode-to-ground short
29	-	<pre>cathode_cathode = 3,</pre>	, // cathode-to-cathode short
30	-	other = 4	// other
31	-	};	
25	+	enum class cell_status	s {
26	+	good = 0,	// good working cell
27	+	dead = 1,	<pre>// dead cells with dead anode</pre>
28	+	<pre>cathode_ground = 2,</pre>	// cathode-to-ground short
29	+	<pre>cathode_cathode = 3,</pre>	, // cathode-to-cathode short
30	+	other = 4	// other
31	+	};	

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.

...

ୢୄ୷୶ୖ	×	Review required At least 1 approving review is required by reviewers with write access. Learn more.	Add your review		
		Hide all checks			
		2 in progress, 14 successful, and 1 failing checks			
	~		Inequired	Dotano	
	~	Iinux tests / rhel8-platform-python (pull_request) Successful in 10m		Details	
	~	Iinux tests / clingo (pull_request) Successful in 29m	Required	Details	
	Iinux tests / clingo-cffi (pull_request) Successful in 24m				
	×	codecov/project — 80.15% (-2.44%) compared to 62f8087		Details	
	~	codecov/patch — Coverage not affected when comparing 62f8087ebbef03		Details	
Merging is blocked Merging can be performed automatically with 1 approving review.					
	En	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, Docker, CVMFS
- 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)

Gamma SuperNEMO-DBD / Falaise

Simulation, Reconstruction and Analysis Software for the SuperNEMO Experiment

SuperNEMO-DBD / Falaise

Simulation, Reconstruction and Analysis Software for the SuperNEMO Experiment



Publishing Documentation

An easy way to do this is through <u>GitHub Pages</u>



Finally: Don't neglect Training until the last slide!

- "Documentation" should really be "Documentation and Training"!
 - Projects should point to existing <u>Software Carpentry</u> (and <u>HSF</u>!) lessons and others (e.g. <u>GitHub Help</u>) for the basics rather than write their own
 - Software Carpentry template good starting point for developing material for your project
 - Hands-on tutorials at, e.g. experiment collaboration meetings, can be valuable, though **you may have to fight for a slot**!
 - Link up remote/in-person material as later opportunities can be limited
- Developers should **also** actively seek training/knowledge opportunities
 - Engage with community meetings/courses like today
 - Institute/National <u>Research Software Engineer</u> groups/organizations
 - *Conferences like* <u>**CHEP**</u>, <u>**SC**</u>, though getting funding can be difficult

In Summary

- Primary challenge for small software projects is **developer FTE**, so this must be used **effectively**
- 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 development that returns investment in standard tools
- In short: Invest time in your tools, use them as designed, and let the computer take the strain!

