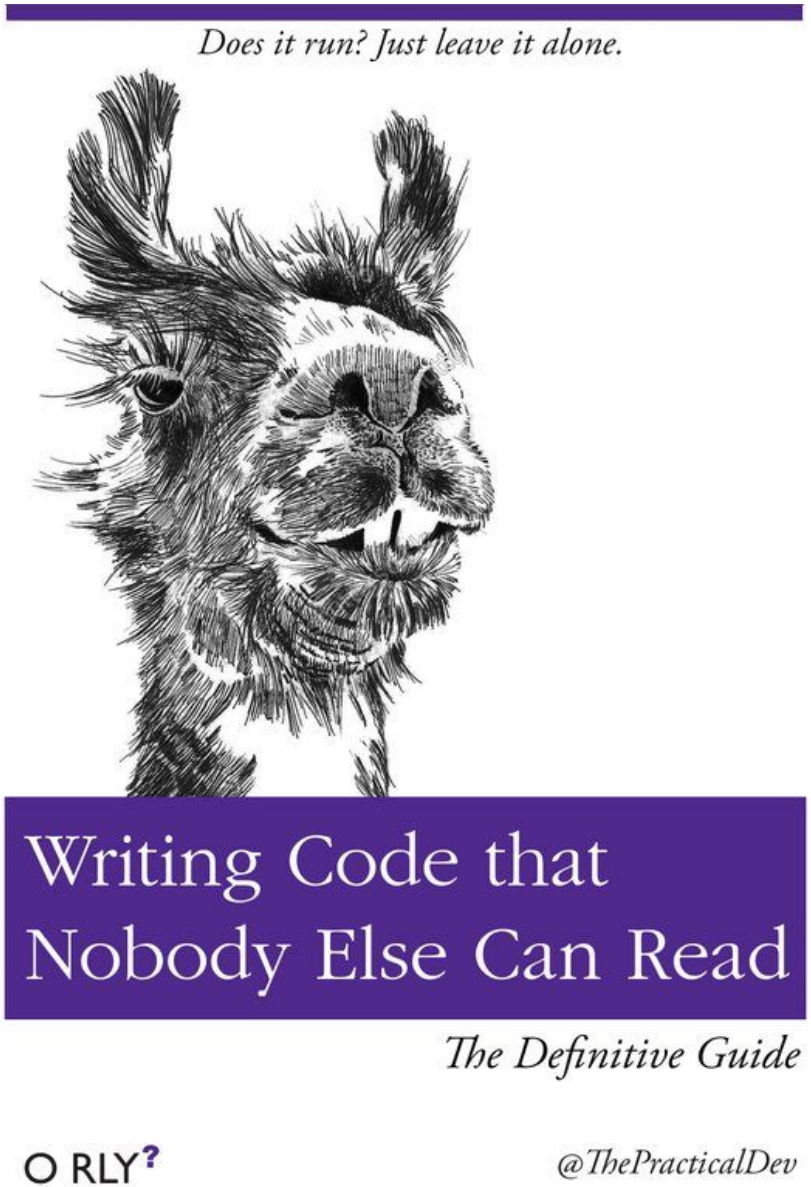
Good software engineering practices for software-writing scientists

Raul P. Pelaez[1]

[1]raulppelaez@gmail.com

Outline

The book


Does it run? Just leave it alone.
Writing Code that Nobody Else Can Read
The Definitive Guide
O RLY?
@ThePracticalDev

You should watch this one

Kenneth Hoste @ FOSDEM26:

How to make package managers scream

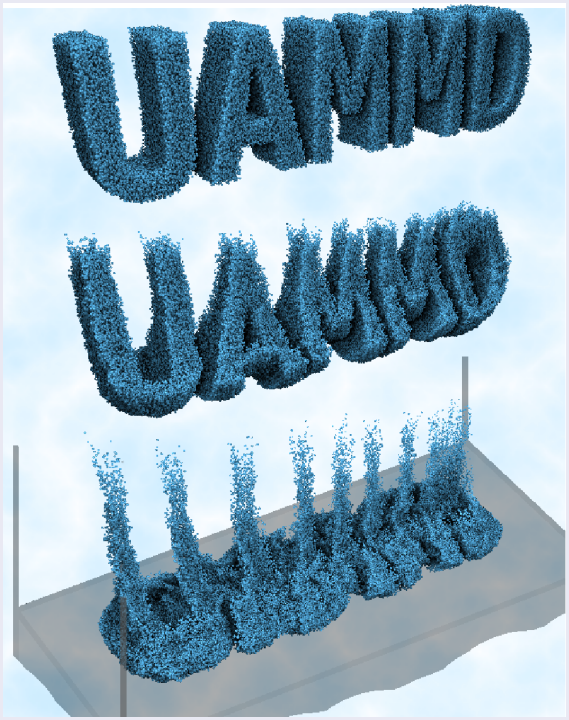
About me

- Ph.D. in Physics: Complex fluid simulation
- GPU hacker
- High-performance computer graphics
- Adjunct Prof. at IE University
- AI Scientist at Mistral Discovery
- AI-enhanced drug-discovery, complex fluids, liquid memory...

Raul in Open Source

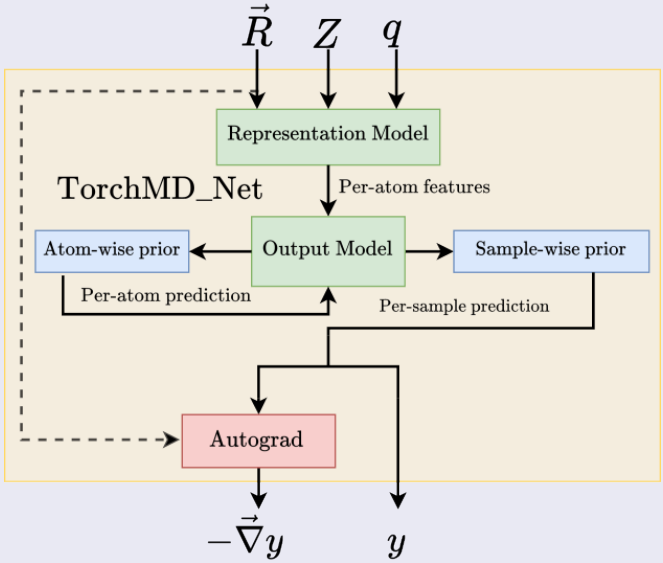
UAMMD

https://github.com/RaulPPelaez/UAMMD



TorchMD-Net

https://github.com/torchmd/torchmd-net



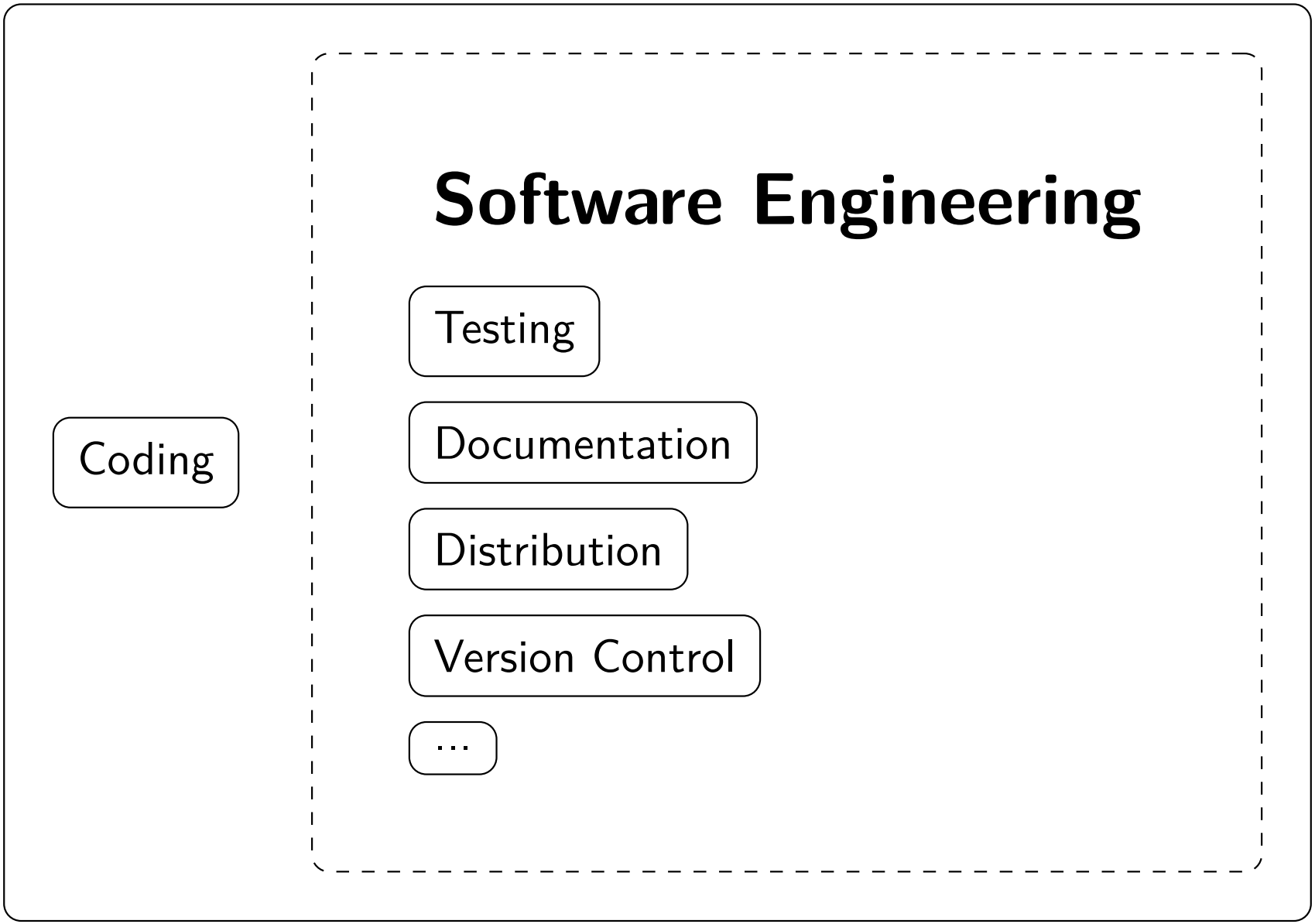
OpenMM

https://github.com/openmm/openmm

LibMobility

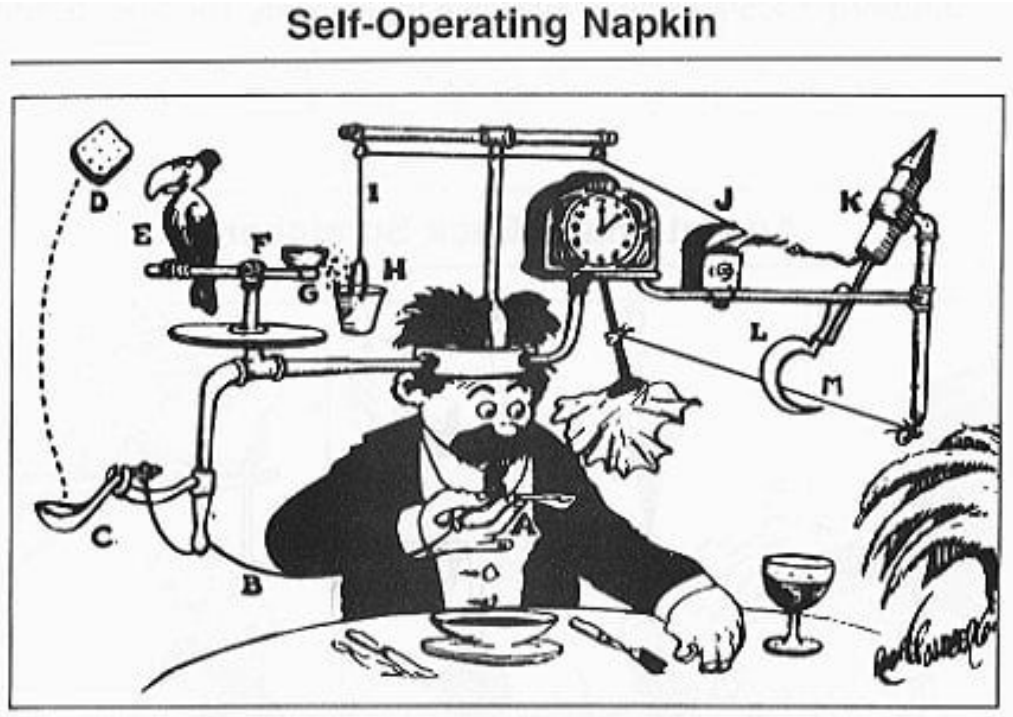
https://github.com/stochasticHydroTools/libmobility

Software development



Complexity:

Anything making a system
hard to understand and modify.



The laws of software development

- Useful programs tend to evolve.
- Change adds complexity.

Lehman, 1974

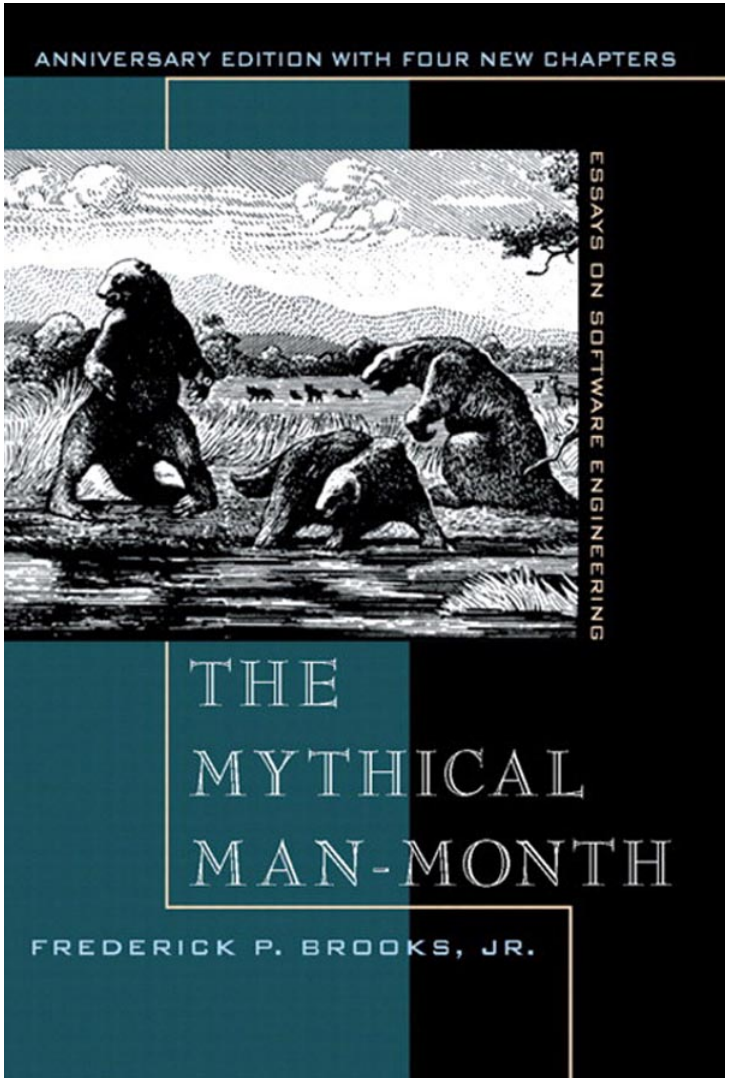
A big ball of mud

A big ball of mud

The softest skill

Failing forwards.

The old new problems



1975

Thou must:

- Thou shalt abide by the **common** rule.

Thou must:

- Thou shalt abide by the **common** rule.
- Thou shalt use **version control**.

Thou must:

- Thou shalt abide by the **common** rule.
- Thou shalt use **version control**.
- Thou shalt **test** thy code

Thou must:

- Thou shalt abide by the **common** rule.
- Thou shalt use **version control**.
- Thou shalt **test** thy code
- Thou shalt **document** thy code

Project structure

Python

Tree

```
Project
|-- project/
|   |-- __init__.py
|   |-- some_script.py
|   |-- ...
|-- tests/
|   |-- test_smth.py
|-- docs/
|-- environment.yml
|-- setup.py # Or
↪   pyproject.toml
|-- .gitignore
```

Project structure

Python

Tree

```
Project
|-- project/
|   |-- __init__.py
|   |-- some_script.py
|   |-- ...
|-- tests/
|   |-- test_smth.py
|-- docs/
|-- environment.yml
|-- setup.py # Or
↪   pyproject.toml
|-- .gitignore
```

Usual steps

```
# Get source
git clone git@github:User/Project
cd Project
# Create environment
conda env create
pip install .     # Install
pytest -v         # Run tests
```

Project structure

C/C++/Fortran/C#/Swift

Tree

```
Project
|-- src/project/
|   |-- some_library.cpp
|   |-- ...
|-- include/project/
|   |-- some_library.h
|-- tests/
|   |-- test_library.cpp
|-- docs/
|-- environment.yml
|-- CMakeLists.txt
|-- .gitignore
```

Project structure

C/C++/Fortran/C#/Swift

Tree

```
Project
|-- src/project/
|   |-- some_library.cpp
|   |-- ...
|-- include/project/
|   |-- some_library.h
|-- tests/
|   |-- test_library.cpp
|-- docs/
|-- environment.yml
|-- CMakeLists.txt
|-- .gitignore
```

Usual steps

```
# Get source
git clone git@github:User/Project
cd Project
# Create environment
conda env create
cmake -B build -DCMAKE_INSTALL_PREFIX=$CONDA_PREFIX
cmake --build build --target install
cd build
ctest -V
```

Standards matter

Make project structure **expectable**

- Dependencies with: conda, pip

Standards matter

Make project structure **expectable**

- Dependencies with: conda, pip
- Build with: pip, CMake

Standards matter

Make project structure **expectable**

- Dependencies with: conda, pip
- Build with: pip, CMake
- Test with: pytest, GTest

Standards matter

Make project structure **expectable**

- Dependencies with: conda, pip
- Build with: pip, CMake
- Test with: pytest, GTest
- Document with: Sphinx, Doxygen

Standards matter

Make project structure **expectable**

- Dependencies with: conda, pip
- Build with: pip, CMake
- Test with: pytest, GTest
- Document with: Sphinx, Doxygen
- Version control with: git

Standards matter

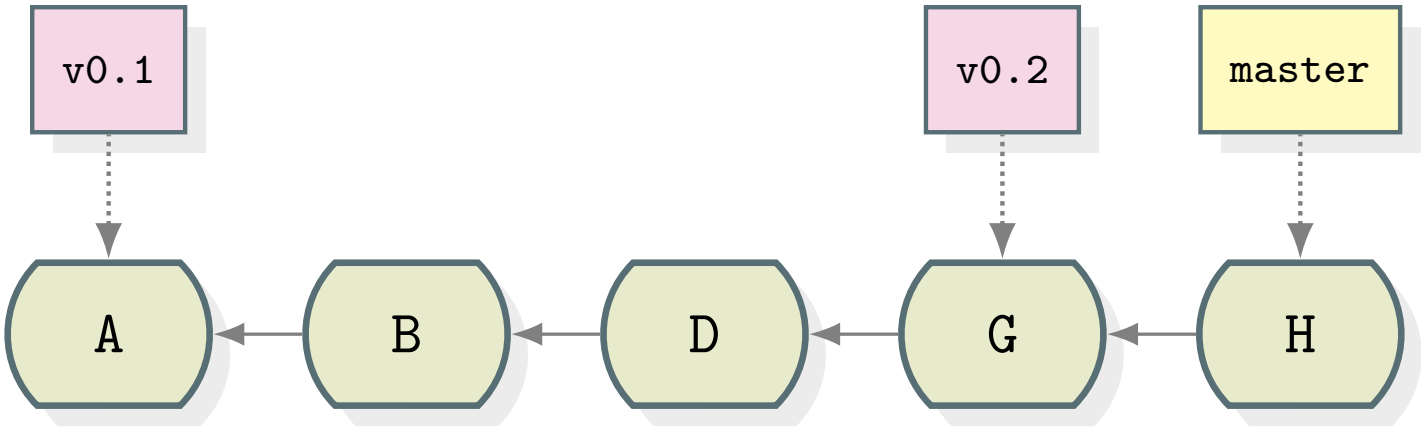
Make project structure **expectable**

- Dependencies with: conda, pip
- Build with: pip, CMake
- Test with: pytest, GTest
- Document with: Sphinx, Doxygen
- Version control with: git
- Format with: black, clang-format

Version control

Use git and GitHub/GitLab/Bitbucket

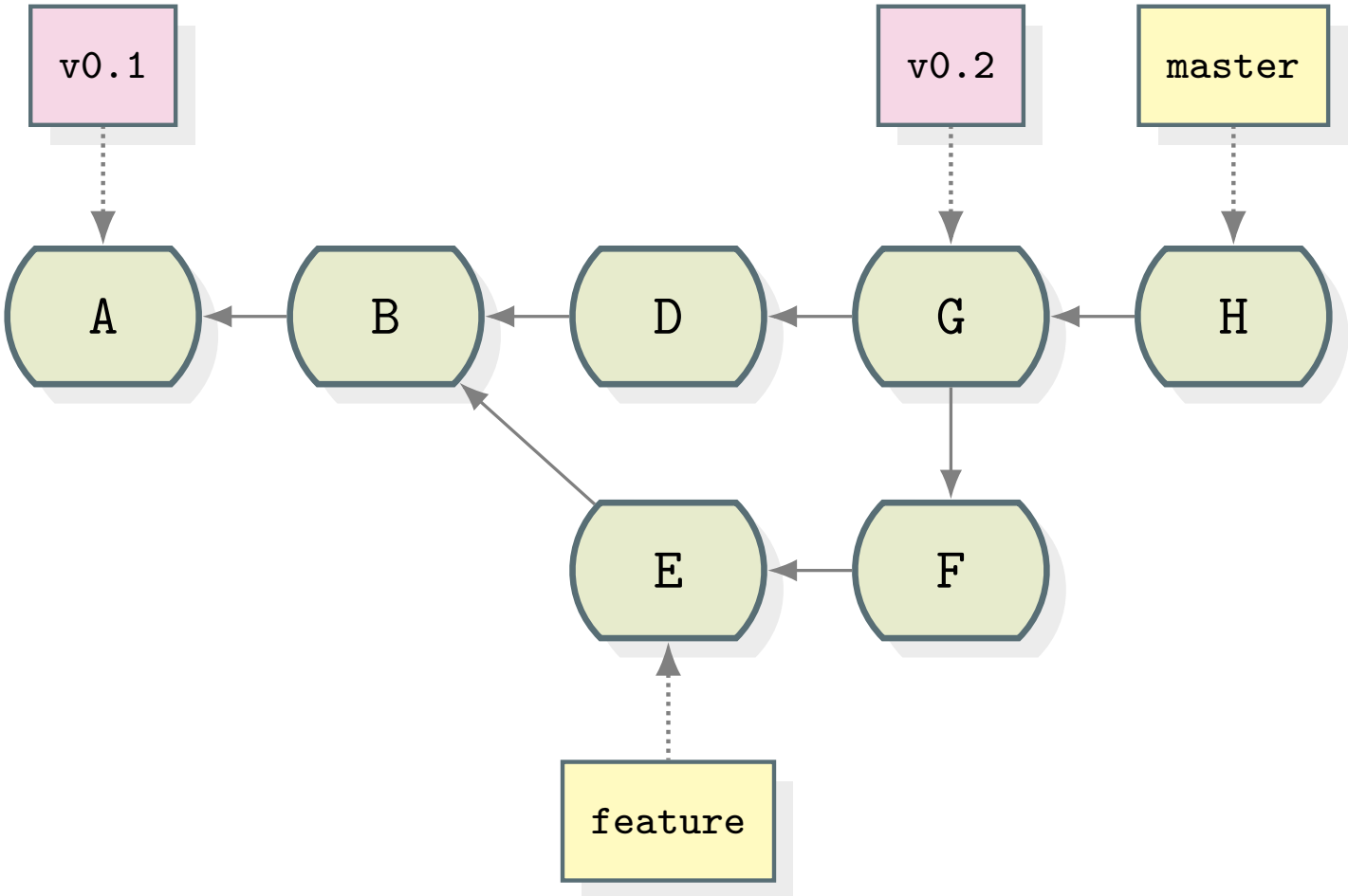
- Know who did what, when and why
- Travel in time



Version control

Use git and GitHub/GitLab/Bitbucket

- Know who did what, when and why
- Travel in time



Version control

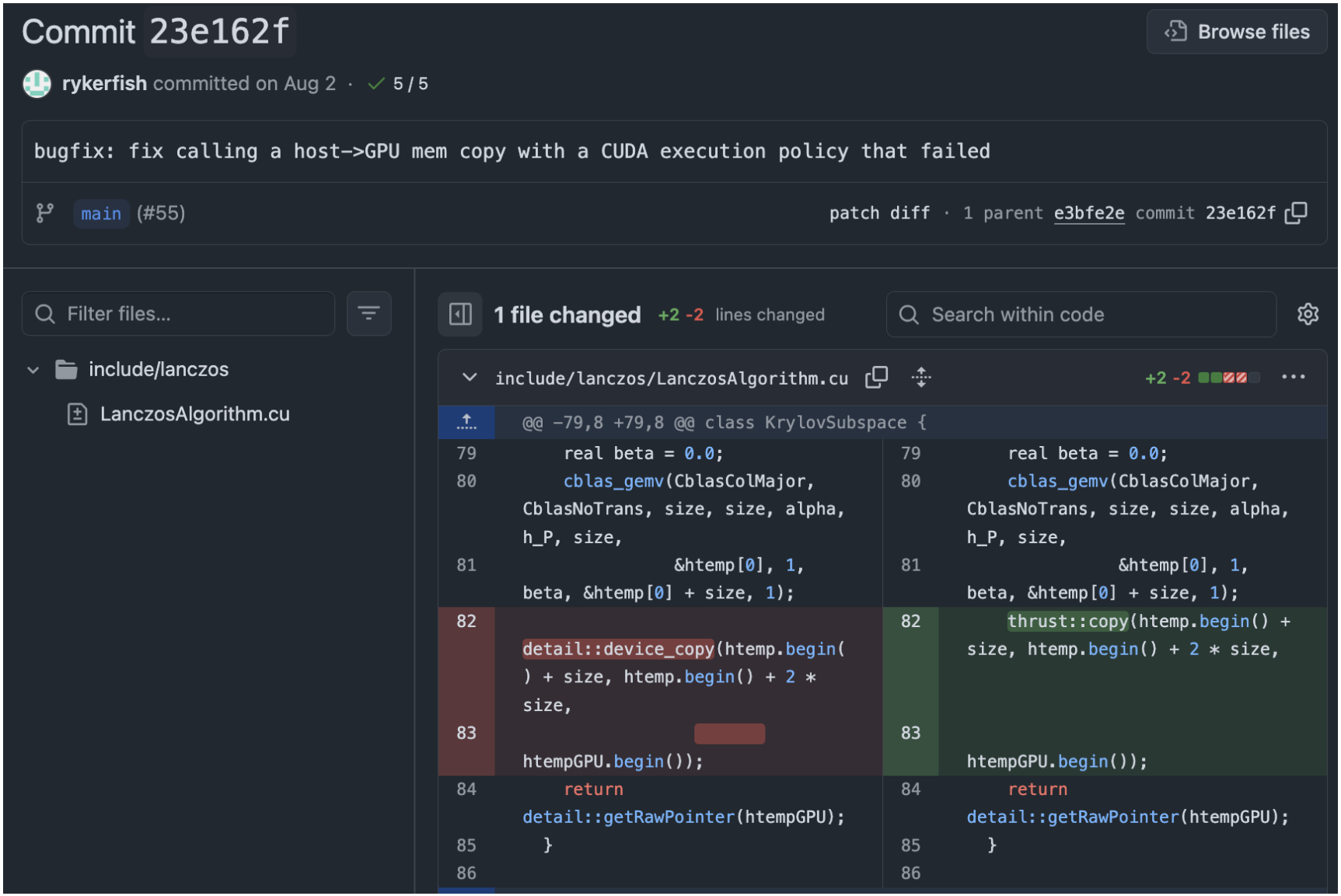
Use git and GitHub/GitLab/Bitbucket

- Know who did what, when and why
- Travel in time

```
3c4a1c6 * | 0.9.1 origin/main build: bump uammd version                    Raul P. Pelaez      1 month
7a547ea * | Fix PSE not cleaning its resources at destruction (#56)        Raul                1 month
4728a5c * | docs: update new solver instructions                          Raul P. Pelaez      1 month
e5a13d0 | * style: formatting                                             Raul P. Pelaez      1 month
e4bd24c | * feat: add an optional callback function for lanczos           Raul P. Pelaez      1 month
a7c7ec9 | * style: formatting                                             Raul P. Pelaez      1 month
6f744fb | * feat: remove dead code                                        Raul P. Pelaez      1 month
08a813b | * feat: compile lanczos in CUDA mode                           Raul P. Pelaez      1 month
7c5fb56 | * feat: make Lanczos part of the repo                          Raul P. Pelaez      1 month
        |/
0ec96ed | * origin/clean_pse build: add the BUILD_JOBS option to setup.py Raul P. Pelaez      1 month
4a8c593 | * build: fix setup.py ignoring the number of cores             Raul P. Pelaez      1 month
c7012d5 | * style: format                                                Raul P. Pelaez      1 month
f9a55b4 | * fix: potential null pointer derreference                     Raul P. Pelaez      1 month
26843ae | * fix: pse not cleaning its resources at destruction           Raul P. Pelaez      1 month
        |/
ce6072f *   Merge pull request #53 from stochasticHydroTools/docs_periodicity_fix  Raul      1 month
        |\
ecd1606 | * format                                                        Ryker Fish          2 months
ab835bf | * get rid of repeated initialization string w/ incorrect information in Ryker Fish  2 months
5ac5788 * | Use CMAKE_CUDA_ARCHITECTURES (#54)                           Ryker               2 months
        |/
73f6e0b * build: add cuda-version to host                               Raul P. Pelaez      2 months
9c32369 * Fix conda package build (#52)                                 Raul                2 months
```

Version control

Use git and GitHub/GitLab/Bitbucket



Version control

Use git and GitHub/GitLab/Bitbucket

- Protect the main branch
- Every change via a Pull Request (PR)

Examples

- https://github.com/stochasticHydroTools/libMobility/pull/57
- https://github.com/conda-forge/staged-recipes/pull/29739

Documentation

- Mostly automatic
- Comprehensive
- Provide an API reference

Examples

- https://uammd.readthedocs.io
- https://torchmd-net.readthedocs.io
- https://docs.openmm.org
- https://spreadinterp.readthedocs.io

Automatic documentation

Integrated in Python

```
def process(data, threshold=0.5):
    results = []
    for item in data:
        if item > threshold:
            results.append(item * 2)
    return results
```

Automatic documentation

Integrated in Python

```python
from typing import List, Iterable

def process_data(
    data: Iterable[float],
    threshold: float = 0.5
) -> List[float]:
    """Return values above a threshold multiplied by 2.

      Args:
          data: An iterable of numeric values.
          threshold: A numeric threshold value.
      Returns:
          A list with the processed values.
    """
    if not isinstance(threshold, (int, float)):
        raise TypeError("Threshold must be numeric")

    return [item * 2 for item in data if item > threshold]
```

Automatic documentation

Also in compiled languages (Doxygen)

```cpp
#include <vector>
/**
  * @brief Process data by multiplying values above a threshold by 2.
  *
  * A very detailed description ...
  *
  * @param data A vector of numeric values.
  * @param threshold A numeric threshold value (default is 0.5).
  * @return A vector with the processed values.
  * @throws std::invalid_argument if threshold is not a number.
  */
std::vector<double> process_data(
    const std::vector<double>& data,
    double threshold = 0.5
) {
  /* ...  */
 return results;
}
```

Examples

The code here:

https://github.com/torchmd/torchmd-net/blob/main/torchmdnet/models/utils.py#L120-L183

Renders as:

https://torchmd-net.readthedocs.io/en/latest/generated/torchmdnet.models.utils.html#torchmdnet.models.utils.OptimizedDistance

Testing

Python uses **pytest**

```
# tests/test_something.py
def test_something():
    assert 1 + 1 == 2
```

Python uses **pytest**

```python
# tests/test_module1.py
from mypackage.module1 import add
def test_addition():
    assert add(1, 2) == 3
    assert add(-1, 1) == 0
    assert add(0, 0) == 0
    assert add(1.5, 2.5) == 4.0
    assert add(-1.5, -2.5) == -4.0
```

C++ uses **GTest**

```cpp
// tests/test_module1.cpp
#include <gtest/gtest.h>
#include "module1.h"
using module1::add;
TEST(AdditionTest, HandlesPositiveNumbers) {
    EXPECT_EQ(add(1, 2), 3);
    EXPECT_EQ(add(10, 20), 30);
}
TEST(AdditionTest, HandlesNegativeNumbers) {
    EXPECT_EQ(add(-1, -1), -2);
    EXPECT_EQ(add(-5, -3), -8);
}
```

Test-Driven Development

- Write a test that fails
- Write code to make it pass
- Refactor
- Repeat

Small repositories

- C https://github.com/RaulPPelaez/c-template-project
- C++ https://github.com/RaulPPelaez/cycles
- Python/C++/CUDA
  https://github.com/RaulPPelaez/spreadinterp
- Python
  https://github.com/RaulPPelaez/orbitals-python

Project structure

Tree

```
Project
|-- project/
|   |-- __init__.py
|   |-- some_script.py
|   |-- ...
|-- tests/
|   |-- test_smth.py
|-- docs/
|-- environment.yml
|-- setup.py # Or
↪   pyproject.toml
|-- .gitignore
```

Project structure

Tree

```
Project
|-- project/
|   |-- __init__.py
|   |-- some_script.py
|   |-- ...
|-- tests/
|   |-- test_smth.py
|-- docs/
|-- environment.yml
|-- setup.py # Or
↪   pyproject.toml
|-- .gitignore
```

Usual steps

```
# Get source
git clone git@github:User/Project
cd Project
# Create environment
conda env create
pip install .     # Install
pytest -v         # Run tests
```

Project structure

```yaml
name: project
channels:
  - conda-forge
dependencies:
  - python
  - pip
  - pytest
  - numpy
```

`environment.yml`: What do I need?

Project structure

```
from setuptools import setup, find_packages
setup(
  name='project',
  version='0.1',
  packages=find_packages()
)
```

`setup.py`: Where is your code?

Project structure

```
*build*
*cache*
*egg-info*
```

`.gitignore`: What is particular to my computer?