Research Software and its Developers: Random thoughts

Bálint Aradi

https://github.com/aradi/

Universität Bremen

BCCMS
Bremen Center for Computational Materials Science

5th conference for Research Software Engineering in Germany

Karlsruhe, 2025-02-25

Introduction

https://en.wikipedia.org/wiki/Town_Musicians_of_Bremen

(Grimm Brüder – Die Bremer Stadtmusikanten)

Bremen Center for Computational Materials Science (BCCMS)

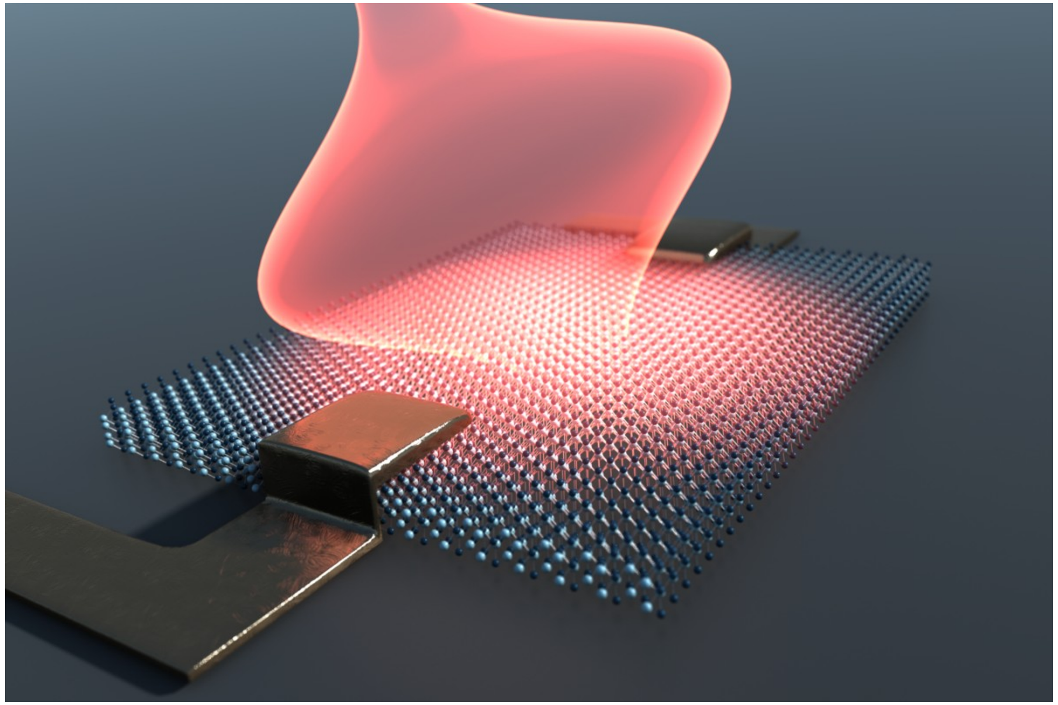
BCCMS
Bremen Center for Computational Materials Science

https://www.uni-bremen.de/bccms

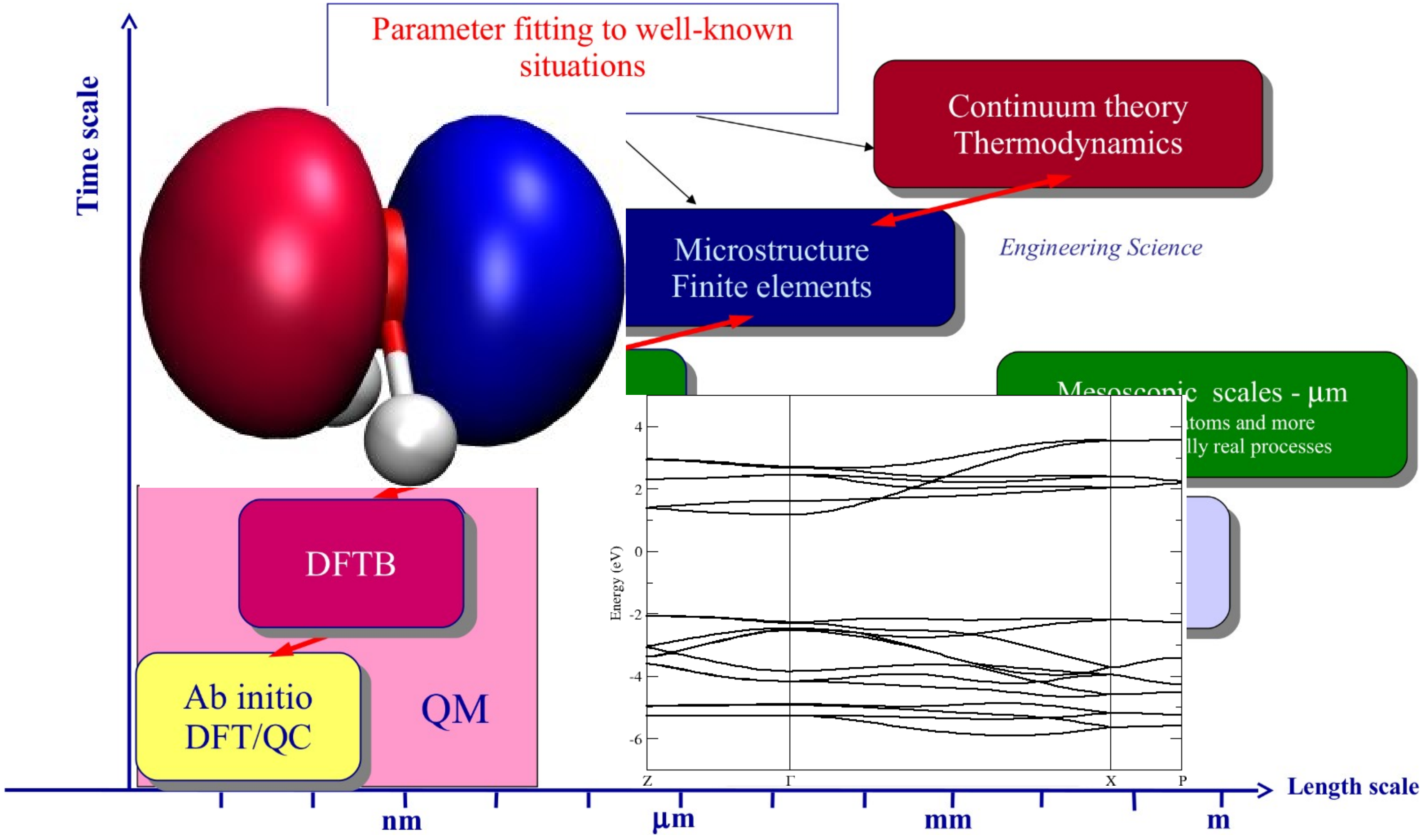
- 8 research groups
- Materials science through computer simulations
- Multi-disciplinary (Physics, Chemistry & Engineering)
- Various aspects of materials science
- Multi-scale (atomistic, meso, macro)

LMCQM

Light-Matter Control of Quantum Materials



My “RSE” background



https://www.dftbplus.org/

- Open source (LGPL)
- First commit 2004
- Modern Fortran (2008/2018)
- ~ 130,000 lines
- 2 core developers / project managers
- 5 – 6 core contributors
- 20 – 30 casual contributors
- Few hundred users

RESEARCH ARTICLE | MARCH 23 2020

DFTB+, a software package for efficient approximate density functional theory based atomistic simulations

Special Collection: Chemical Physics Software Collection , Electronic Structure Software

B. Hourahine; B. Aradi; V. Blum; F. Bonafé; A. Buccheri; C. Camacho; C. Cevallos; M. Y. Deshaye; T. Dumitrică; A. Dominguez; S. Ehlert; M. Elstner; T. van der Heide; J. Hermann; S. Irle; J. J. Kranz; C. Köhler; T. Kowalczyk; T. Kubař; I. S. Lee; V. Lutsker; R. J. Maurer; S. K. Min; I. Mitchell; C. Negre; T. A. Niehaus; A. M. N. Niklasson; A. J. Page; A. Pecchia; G. Penazzi; M. P. Persson; J. Řezáč; C. G. Sánchez; M. Sternberg; M. Stöhr; F. Stuckenberg; A. Tkatchenko; V. W.-z. Yu; T. Frauenheim

Open source engagement

Fortuno – flextensible unit testing framework for Fortran

Fortuno (Fortran Unit Testing Objects) is a flexible & extensible, object-oriented unit testing framework

https://github.com/fortuno-repos/fortuno

Fypp — Python powered Fortran metaprogramming

build passing

Fypp is a Python powered preprocessor. It can be used for any programming languages but its primary aim is

https://github.com/aradi/fypp

HSD — Make your structured data human friendly

Utilities to read and write files in the Human-friendly Structured Data (HSD) format.

https://github.com/aradi/hsd-python

Outline

- Roles and competencies of research engineers
- Where/When should RSEs enter scientific projects
- Strengthening the RSE-skills of scientists
- Thoughs about software reuse and library approach
- Building communities: the example of the Fortran community
- Final thoughts

Who is a Research Software Engineer?

Create an image of a research software engineer at work

Here's the image of a research software engineer at work. You can see her focusing on her tasks amidst a busy and intellectual workspace.

RSE as depicted by DALL-E

Who is a Research Software Engineer? (#2)

A **Research Software Engineer** (**RSE**) is a professional who combines expertise in **software development** with **research methodologies** to create, optimize, and maintain software used in academic and scientific research. RSEs ensure that computational tools, models, and workflows are efficient, reproducible, and sustainable, bridging the gap between **research** and **high-quality software** engineering.

RSE as defined by ChatGPT 4o

Key characteristics

- **Hybrid role**
  Software engineering skills with *domain-specific research knowledge*
- **Focus on sustainability**
  Develops software that is *maintainable*, *scalable* and *reusable*
- **Reproducibility advocate**
  Implements *best practices* to ensure research results can be replicated
- **Collaboration-oriented**
  Works closely with researchers, scientists and developers.
- **Innovation driven**
  Applies *cutting-edge technologies* (e.g., HPC, AI, cloud computing) to research problems.

Functional RSE competencies (as defined by teachingRSE)

Classical software engineer skills

Adapting to the software life cycle (**SWLC**)

Creating documented code building blocks (**DOCBB**)

Building distributable software (**DIST**)

Use software repositories (**SW**)

Software behaviour awareness and analysis (**MOD**)

Conducting and leading research (**NEW**)

Understanding research cycle (**RC**)

Software reuse (**SRU**)

Software publication & citation (**SP**)

Use domain repositories/directories (**DOMREP**)

Working in a team (**TEAM**)

Teaching (**TEACH**)

Project management (**PM**)

Interaction with users and other stakeholders (**USERS**)

F. Goth et al., Foundational Competencies and Responsibilities of a Research Software Engineer

https://github.com/the-teachingRSE-project/competencies

Research Software Engineers: ideal

Research software engineer of different genders resembling super heros (by DALL-E)

Research Software Engineer: doing the split

Conducting and *leading* research (**NEW**)

Software engineer skills

Conducting, especially **leading research** is already a **full time job**

- Following development of research field
- Visiting conferences
- Writing applications for research support
- Supervising BSc. / MSc. / PhD-students

University positions for RSEs in Germany

- Typically “Mittelbau” (E12/E13)
- “Leading research” is (usually) not part of the job description

Domain specific knowledge

Domain-specific knowledge

Fundamental theories

- Core scientific theories relevant to the domain (e.g. QM, fluid mechanics, thermodynamics, etc.)

Mathematical proficiency

- Advanced mathematical skills typically used in the domain (linear algebra, calculus, etc.)
- Deep understanding of applied numerical methods and their limitations

Project related knowledge

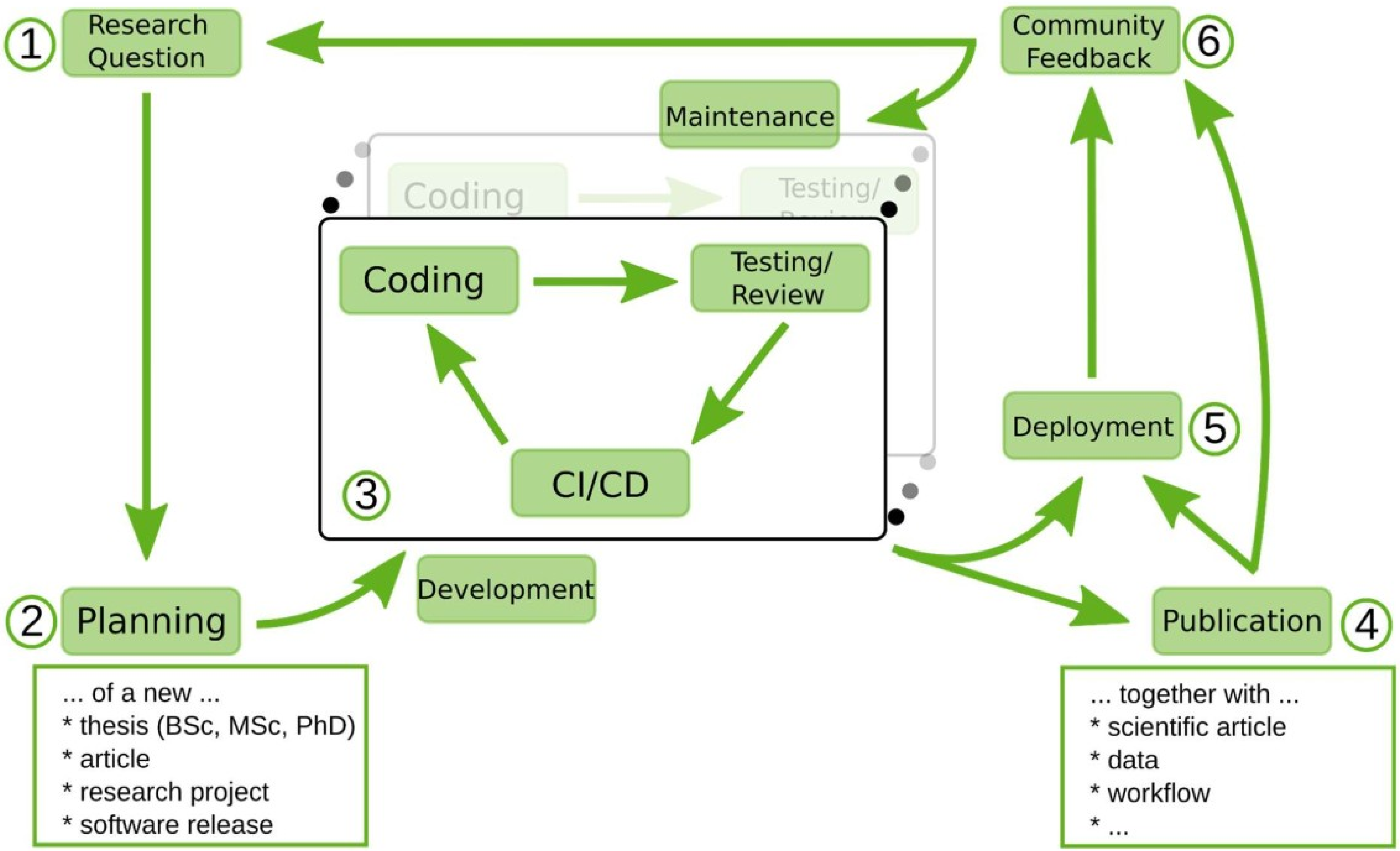
- Goal of the project
- Motivation of the implemented equations
- Connecting numerics with on physical interpretation

Having a science educational background can be of great advantage

- Specialization to RS-developer in normal science curriculum should be possible
- Bachelor/Master thesis with strong numerical focus

Where/When does the RSE enter the scientific software project?

The research project life cycle



https://everse.software/RSQKit/life_cycle

Classification of research software

RSQKit three tire model

Analysis code

- Personal use, small scope
- captures computational research processes and methodology

Prototype tools

- Demonstrating new idea
- Developed & used by more than one person

Research software infrastructure

- Broadly applicable
- Large, distributed development team

https://everse.software/RSQKit/three_tier_view

DLR software classification

Application Class 0

- Personal use, small scope
- Distribution outside of organization not planned

Application Class 1

- Non-developers can use
- Extension/development by externals possible

T

Application Class 2

- Defined development process
- Long-term development and maintenance

Application Class 3

- Critical software
- Software with product characteristics

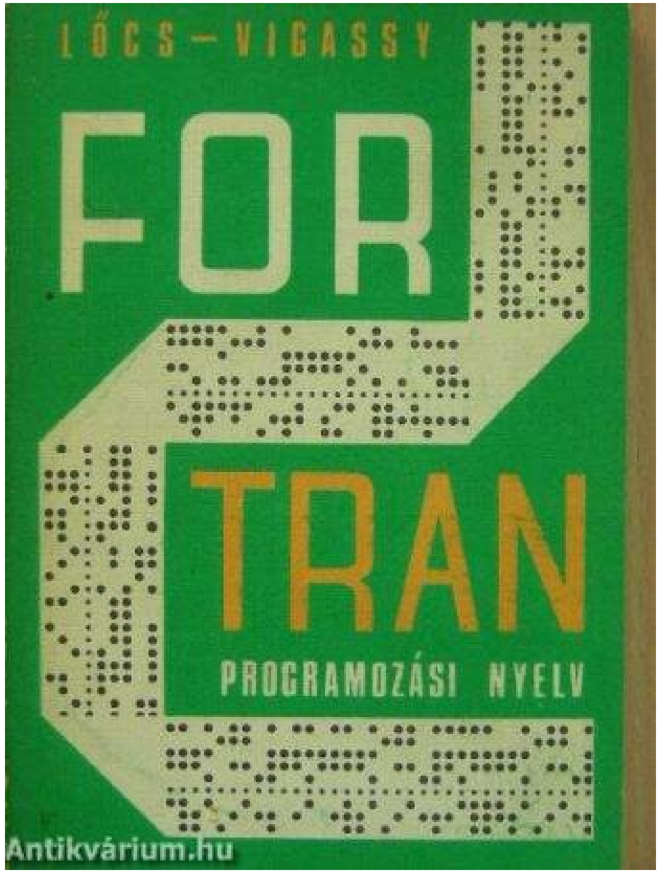
P

DLR Software Engineering Guidelines (v1.0.0)

It all starts with teaching

Here is the comic-style image depicting the theme "It all starts with teaching," featuring a diverse group of children with their teacher in a vibrant classroom. (DALL-E)

"Teaching RSE-skills" around 1997



Published in 1972

Note: punch cards and **fixed format source** were **not state-of-the-art** for long time any more (Fortran 90/95 standard published already)

Teaching ALL science students the basic skills

Basic skills

- Turning (mathematical) problems into algorithms
- Project work, interaction

Work-flow basics

- Building the program (if applicable)
- Testing
- Code quality analysis
- Version control (collaborative development)
- Packaging and distribution

Language basics

- Basic data types
- (Minimal knowledge about numerical arithmetic)
- Basic containers / arrays
- Functions, modules, etc.
- File I/O
- Visualization
- Error handling (exception & co.)
- Basics of relevant programming paradigms (functional programming, OOP, etc.)

Example: (quasi) introductory course for physics students

Disclamer: this is my course @ Uni Bremen

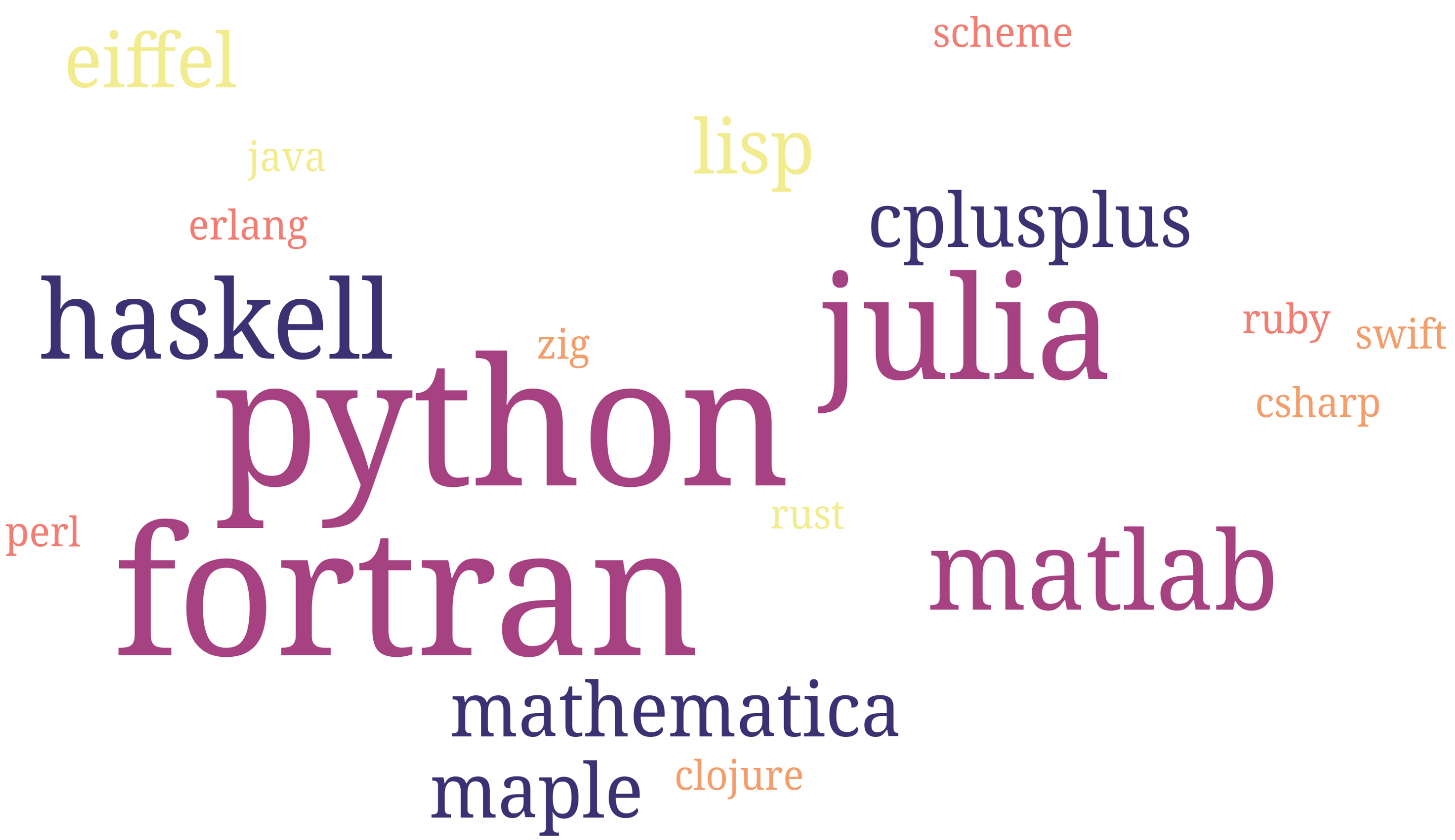
Lecture plan

0. Setting up the working environment
1. Python basics
2. Tuples and lists
3. Sets and dictionaries
4. Functions and arrays
5. File I/O and plotting
6. Git basics and Python modules
8. Further Git features
9. Unit testing and code quality analysis
10. Parallel and collaborative development with Git
11. Command line arguments, packaging and distribution

https://atticlectures.net/scipro/python-2024/

- Introductory course (2 SWS)
- **Inverted class room** concept
  - **Learning @ HOME**:
    - tutorial videos, detailed slides
    - cheat sheets
  - **Programming @ UNI**:
    - programming with live support (generalized pair programming)
  - **Final project** (~400 lines)
    - Numerical simulation software
    - Collaborative project (2 students, via Git)
    - Tests
    - Documentation
    - Packaging

Which (1^{st}) language should we teach to scientists?



Which language should we teach? (#2)

```
#include <iostream>
#include <sycl/sycl.hpp>
using namespace sycl;
const std::string secret{"Ifmmp-!..."};
const auto sz = secret.size();
int main() {
    queue q;
    char* result = malloc_shared<char>(sz, q);
    std::memcpy(result, secret.data(), sz);
    q.parallel_for(sz, [=](auto& i) {
        result[i] -= 1;
    }).wait();
    std::cout << result << "\n";
    free(result, q);
    return 0;
}
```

Reinders et al., Data Parallel C++, 2nd ed.

Which language should we teach? (#3)

```
program hello
  implicit none

  character(*), parameter :: secret = "fjdsflks81824fsdn,mnj3..."
  character(:), allocatable :: res
  integer :: i

  res = secret
  do concurrent (i = 1 : len(secret))
    res(i:i) = char(ichar(res(i:i)) - 1)
  end do
  print *, res

end program hello
```

At the interface between R and SE

Research

- Physical model
- Mathematical formulation of the model
- Algorithmic description of the math

Software engineering

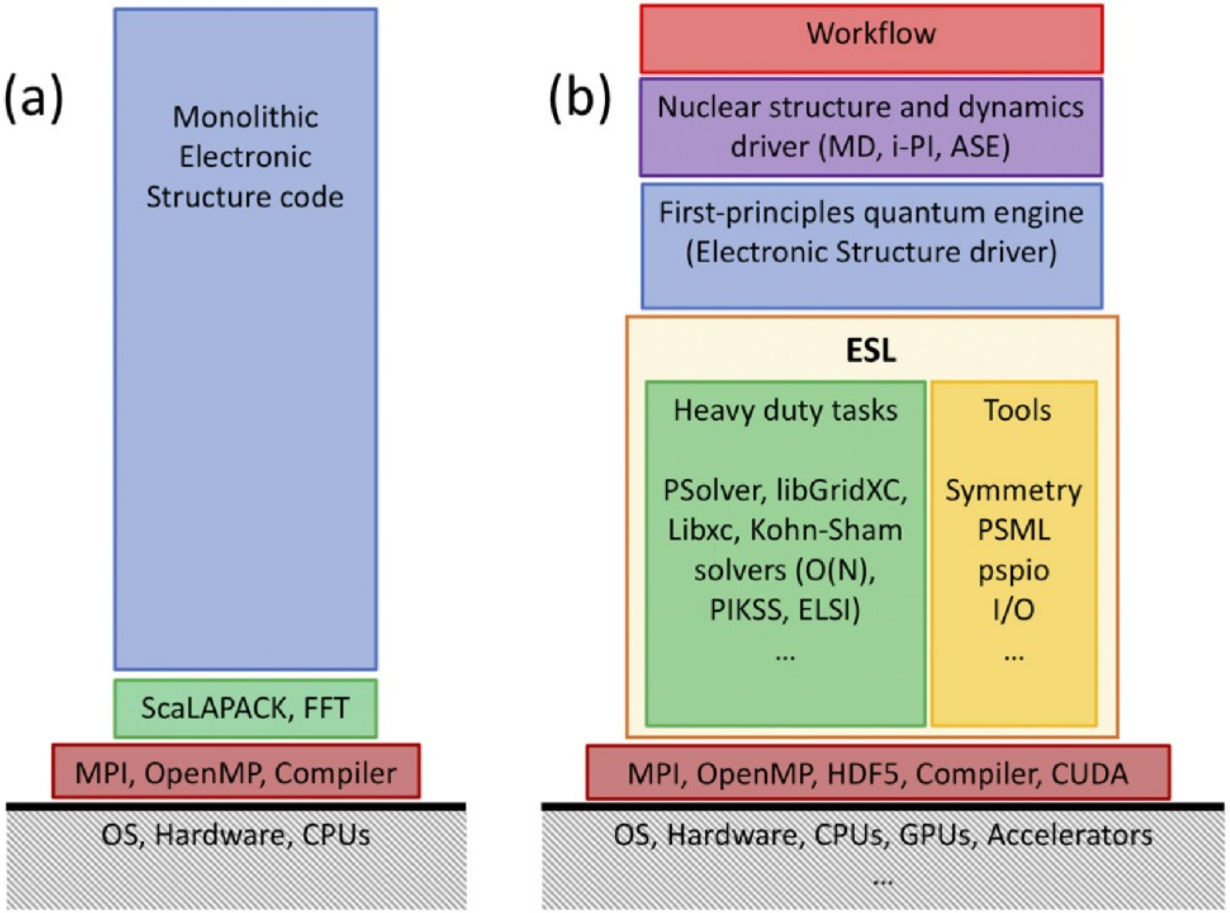
- Numerical modelling of the algorithm
- Technical details of the implementation
- Technology know-how

```
if (this_image() == 1) then
   n_circle = in_circle
   do i = 2, num_images()
      n_circle = n_circle + in_circle[i]
   end do
   ...
end if
```

Co-array Fortran

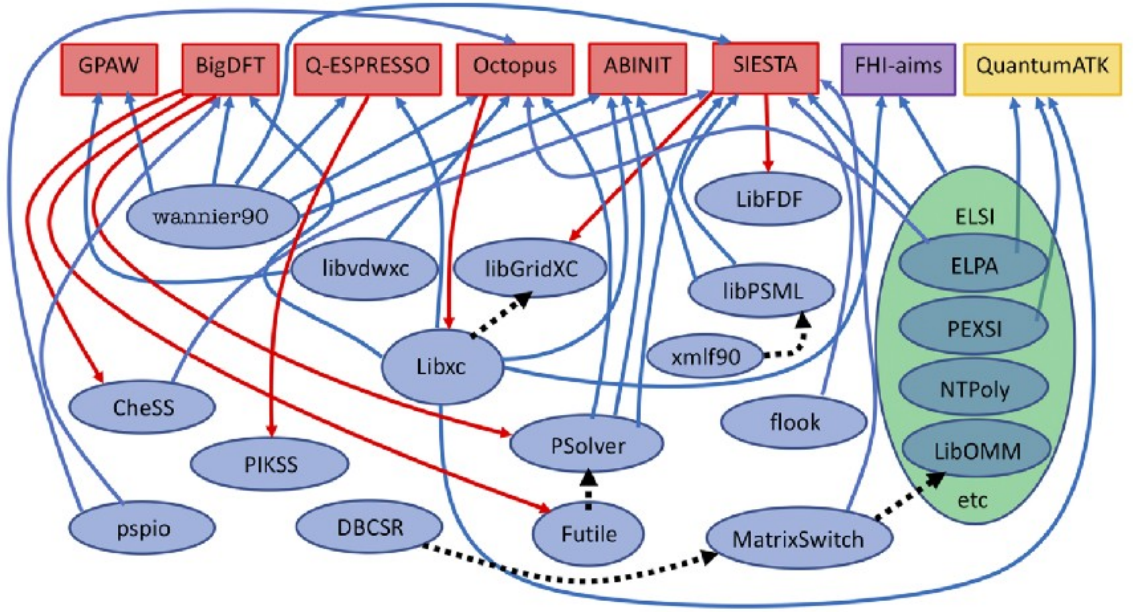
Software reusage example: ESL (break existing monolithism)

https://esl.cecam.org



M. J. T. Oliveira et al, J. Chem. Phys 153, 024117 (2020)

- Started 2014 as a CECAM initiative
- Community-maintained library of software
- Lower barrier for new el. struct. codes
- Identifying existing libraries to include
- Extract/recode sub-packages as libraries
- Incorporate libraries into participating codes



Exchange between developers on best practices

Dependencies

- **MpiFx**: Modern user friendly Fortran-wrappers around MPI
- **ScalapackFx**: Moder user friendly Fortran-wrappers around ScaLAPACK

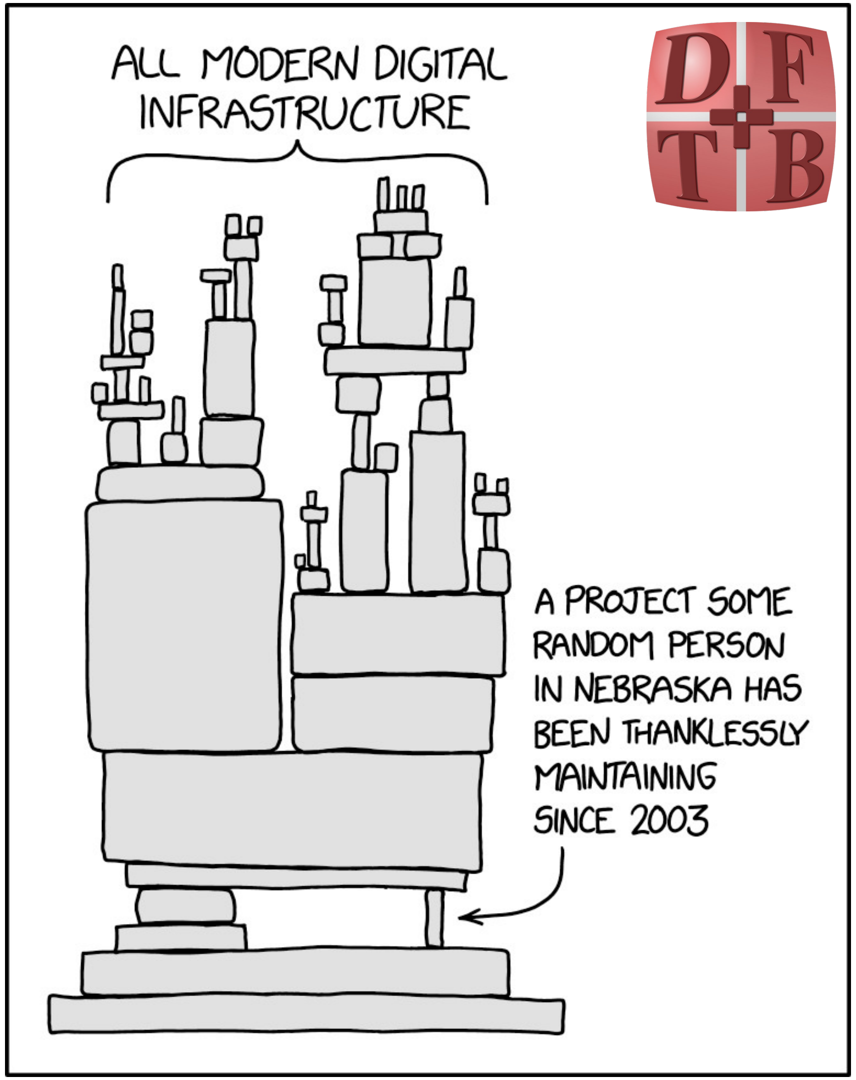
Class 0

- **LAPACK**, **BLAS**, **ScaLAPACK**
- **ARPACK-NG, MAGMA**

Class 1

Class 2

- **ELSI**: parallel diagonalizers and solvers (depends itself on 6-7 further scientific packages)
- **libNEGF**: Non-Equlibrium
- **Simple-DFTD3**: offering D3-dispersion correction
- **TBLite**: tight binding with xTB-Hamiltonian model
- **Plumed**: Meta-dynamics driver
- **ChIMES-calculator**: Chebyshev-polynomial base force fields



Someday ImageMagick will break for good and we'll have a long period of scrambling as we try to reassemble civilization from the rubble

https://xkcd.com/2347/

Levels of (scientific) projects dependencies

Class 0
Project under your control

- Project developed by the same group/community as the depender
- Expected life-time similar to depender’s
- Chances of unexpected API-changes minimal
- Packaging / distribution strategy similar to depender’s

Class 1
Standard project

- Project developed by a big community
- De-facto standard library for many projects
- Expected life-time beyond depender’s
- Chances of unexpected API-changes minimal
- Packaging / distribution strategy likely to be compatible with depender’s

Class 2
Project with uncertain future

- Project developed by a small community
- Uncertain life-time expectancy
- Unexpected API-changes possible
- Packaging / distribution strategy might be incompatible with depender’s
- Dependency should be **optional**

How we deal with class 2 (optional) dependencies

Meta-programming approach

```
type :: TDftbPlus
  #:if WITH_SOCKETS
  type(ipiSocketComm) :: socket
  #:endif
end type

#:if WITH_SOCKETS
  call sendEnergyAndForces(&
      & env, this%socket, ...)
#:endif
```

- \+ Build time error on usage of non-existing component
- \- Code less readable (every usage/import must be guarded)

Mocking approach

```
#:if not WITH_SOCKETS
type :: ipiSocketComm
   ..
contains
  procedure :: ...
end type ipiSocketComm
#:endif
```

- \+ Requires minimal amount (or no) meta-programming
- \- Might be tedious for complex objects/interfaces
- \- Incorrect usage of non-existing component might be detectable only at run-time

Packaging & distribution: for whom?

Novice / casual user

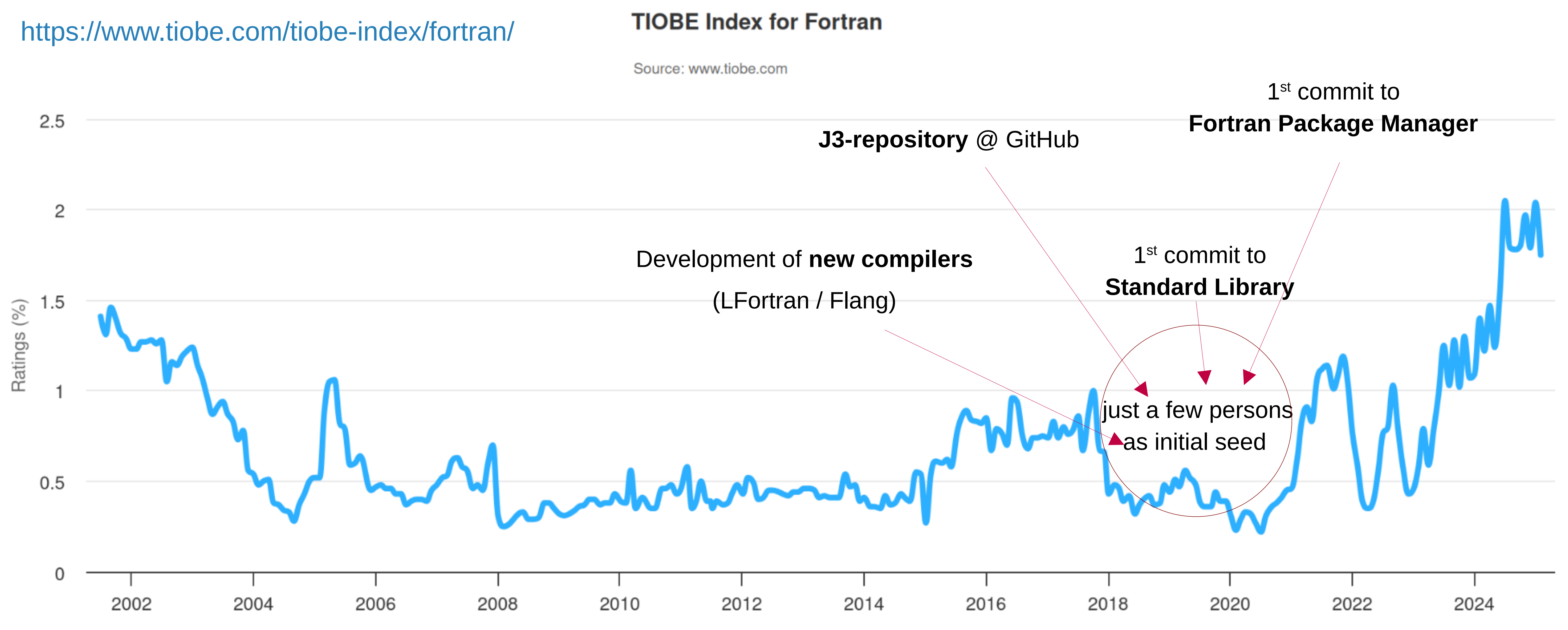
- Needs lowest possible entry barrier
- Out-of-the-box (no need of fine-tuning)
- Target: Laptop of the user
- Binary packages preferable
- deb, rpm, homebrew, **conda**, …

Professional user

- Can deal with more complex builds
- Fine-tuning required
- Target: Workstation, HPC environment
- Reproducible build
- easy-build, spack, ...

Building of communities: the revival of Fortran

https://www.tiobe.com/tiobe-index/fortran/



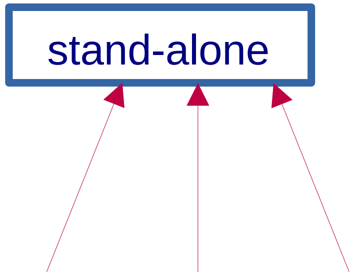
- Lowering entry barrier
- Making participation “cooler”
- Offering long-term perspective

Open Source Community Building
Wed 16:00

- At which application level do we need a community?

Rewarding contributors / Citation of software products

Stand-alone code



- Citing the original publication (e.g. VASP)

  [1] G. Kresse and J. Hafner, Phys. Rev. B 47 , 558 (**1993**).

  [2] G. Kresse and J. Furthmüller, Comput. Mat. Sci. 6 , 15 (**1996**).

  [3] G. Kresse and J. Furthmüller, Phys. Rev. B 54 , 11 169 (**1996**).

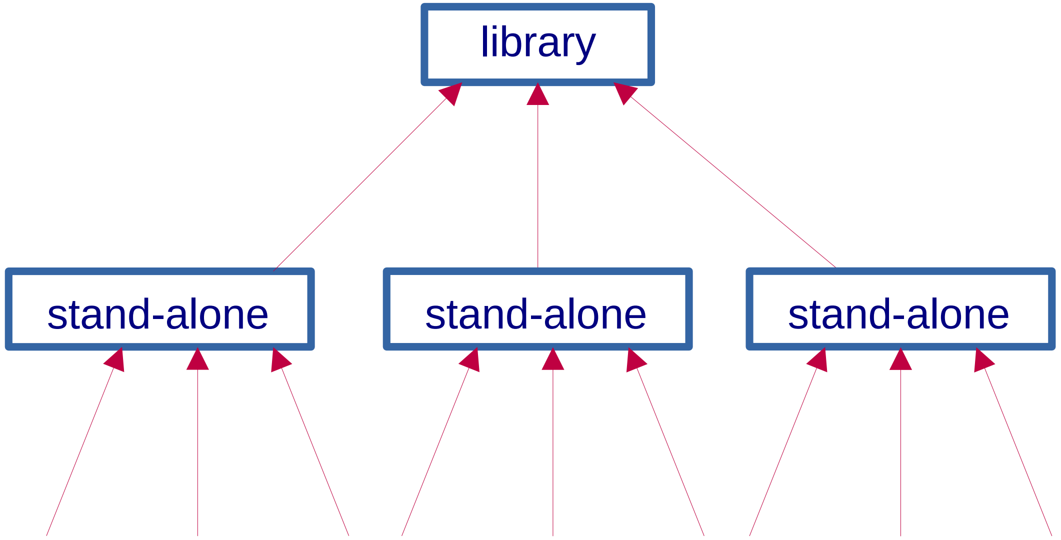
- Refreshing reference publication regularly, Inviting active developers of last period as co-authors (example: DFTB+)

  B. Aradi et al., J. Phys. Chem. A 111, 5678 (**2007**).

  B. Hourahine et al., J. Chem Phys. 152, 124101 (**2020**).

  B. Hourahine et al., J. Phys. Chem. A, submitted (**2025**).

Library



- Essential libraries often orders of magnitude fewer cited as dependent stand-alone codes (example: libxc)
- We need to work on meaningful indirect citation index (or any other quantitative measure for usage)

Academic code development vs. outsourcing to industry

- Domain expertise
- Customizability
- Flexibility & experimentation
- Reproducibility & open science
- Knowledge retention
- Cost efficiency (?)
- Curiosity motivated
- Education & skill development
- Ethical & privacy considerations

RSE problem:

- Lack of RSE best practices
- Lack of proper quality assurance
- Scalability issues
- Slow adoption & usability issues
- Limited user support

System problem:

- Limited user support
- Time constraints
- Limited long-term maintenance
- Funding challenges

- Risk of one person dependence

Hybrid models might bring synergies (example: DFTB+ in Biovia's Materials Studio)