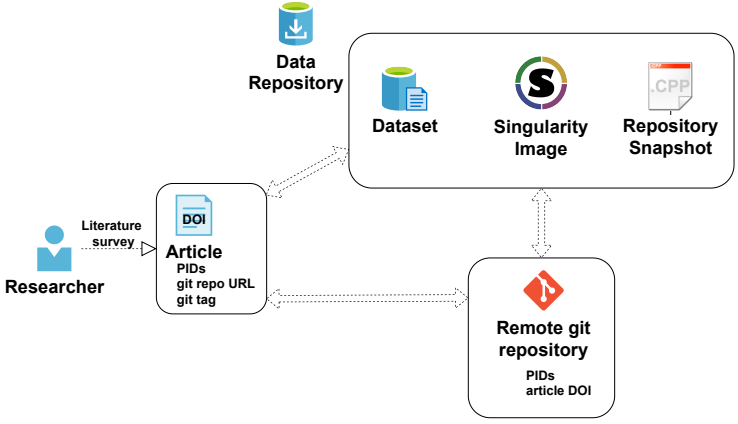
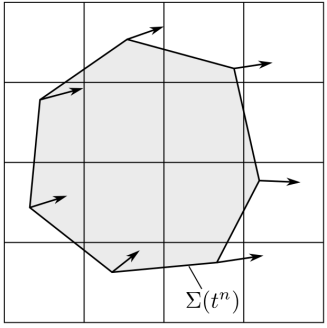
A Workflow for Increasing the Quality of Scientific Software (in Computational Science and Engineering)

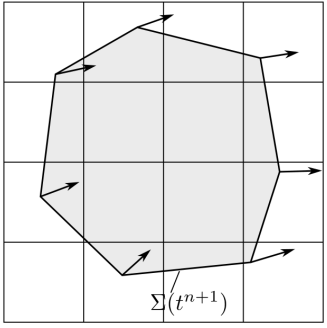
CRC 1194 Z-INF Webinar 2021-04-22

TECHNISCHE UNIVERSITÄT DARMSTADT



Motivation: multiphase flow simulation software





- Fluids that do not mix are separated by an interface $\Sigma(t)$ (surface in 3D).
- Goal: track $\Sigma(t)$ as it moves in time t and changes its topology.

Motivation: multiphase flow simulation software

Lagrangian / Eulerian Interface Advection (LEIA) Methods

TECHNISCHE UNIVERSITÄT DARMSTADT

LEIA methods[1,2,3] require thorough testing:

- Verification cases: evolution of $\Sigma(t)$ and two-phase flows with exact solutions.
- Validation with respect to experiments.
- Serial and parallel computational efficiency.

[1] Marić, T., Marschall, H., & Bothe, D. (2015). lentFoam–A hybrid Level Set/Front Tracking method on unstructured meshes. Computers & Fluids, 113, 20-31.

[2] Tolle, T., Bothe, D., & Marić, T. (2020). SAAMPLE: A Segregated Accuracy-driven Algorithm for Multiphase Pressure-Linked Equations. Computers & Fluids, 200, 104450.

[3] Marić, T., Kothe, D. B., & Bothe, D. (2020). Unstructured un-split geometrical Volume-of-Fluid methods–A review. Journal of Computational Physics, 420, 109695.

Computational Science and Engineering software in university research groups

Boundary and initial conditions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Publish or perish 🎓[4] prioritizes publications over scientific software.
- Dedicated resources for increasing software quality are usually not available.
- Ph.D. students rotate every 4-5 years, postdocs every 1-2 years.
  - Little or no overlap between successors and predecessors.
- Large-scale software design is not a necessary part of the CSE curriculum.
  - Different CSE background: (Applied) Mathematics, Mechanical Engineering, Physics, Informatics.
- Real-world example: onboarding people into OpenFOAM module development.

[4] Symbol of a publish-or-perish simplification of the workflow :)

Computational Science and Engineering software in university research groups

Divergence

TECHNISCHE UNIVERSITÄT DARMSTADT

- Not being able to continue development from an earlier state.
- Reproducing results from a publication is not possible.
  - Data, source code and publication are not archived and cross-linked.
  - The version used to generate the data is not documented.
- Not being able to re-use a model from a publication.
  - The model is not implemented in a modular way.
  - Version integration was not done.
  - Non-granular commits were used.
- Having no overview of the impact of a change on the rest of the module.

A workflow for increasing the quality of scientific CSE software

TECHNISCHE UNIVERSITÄT DARMSTADT

The workflow is using GitLab, specifically TUGitLab. Optional steps are in round () brackets.

1. Track the issues in a Kanban board.
   - Model issues as Progress Tracking Cards[5].
2. Use version-control with a simple branching model.
3. (Apply Test-Driven Development (TDD) for CSE software.)
4. (Enable Continuous Integration with an emphasis on result visualization.)
5. Cross-link software, result data, and report/article when reaching a milestone.
   - When submitting a publication to peer-review.
   - After the publication has been accepted.
   - When giving up on an idea.
6. (Publish a Singularity image with the code and data.)

[5] Developed by Better Scientific Software.

TECHNISCHE UNIVERSITÄT DARMSTADT

The workflow is developed with OpenFOAM projects but it is tested with other software.

Disclaimer: This offering is not approved or endorsed by OpenCFD Limited, producer and distributor of the OpenFOAM software via www.openfoam.com, and owner of the OPENFOAM® and OpenCFD® trade marks.

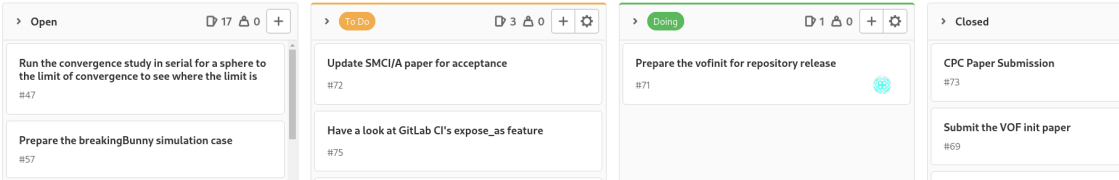
Coordinating work on shared text-based information: different people editing same files.
Tracking the status of tasks, bugs, and ideas in one place.
Avoiding situations like these:

- Why did we try that model a year ago?
- What was that bug in X?
- Meetings for discussing task status.
  - Fine for a Ph.D. student in a single project, not so fine for a postdoc with 5 projects.
  - Not necessary if no input is required, only the status update.

Issue tracking on GitLab Kanban boards

Kanban board at TUGitLab

TECHNISCHE UNIVERSITÄT DARMSTADT



- Create tasks and update their status.
- Great for problems / bugs that can't be solved immediately.
- Descriptions and comments can be done within the tasks (no searching through emails).

Issue tracking on GitLab Kanban boards

Progress Tracking Cards I

TECHNISCHE UNIVERSITÄT DARMSTADT

Progress Tracking Cards[6] are great for teamwork.

1. Title the task that will be solved by the team in terms of what will be achieved.
2. Separate the task into subtasks and mention who is responsible for what.
   - Each sub-task is titled based on the achieved result.
3. Comment in the task, chat on Slack (Discord, ...) if more discussion is necessary, Zoom / meet if even more discussion is necessary.

[6] https://bssw.io/ Better Scientific Software

Software engineering: version control

What is version control?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Management of versions of (usually) textual data, like publications and scientific codes.
- Nowadays version control is **essential** for scientific codes of all shapes and sizes.
- A basis for productive research in teams and increasing the quality of scientific software[7].

[7] Maric, Tomislav, Lehr, Jan-Patrick, Papagiannidis, Ioannis, Lambie, Benjamin, Bothe, Dieter, & Bischof, Christian. (2021, April). A Workflow for Increasing the Quality of Scientific Software (Version 1.0). Zenodo. http://doi.org/10.5281/zenodo.4668439

Software engineering: version control

Why use version control?

TECHNISCHE UNIVERSITÄT DARMSTADT

Why use version control in scientific codes?

- The ability to work with others (colleagues or students) on your research project.
  - Work together faster.
  - Re-use an interpolation method of a colleague in the group.
- The ability to trivially try out new ideas and switch back if they don't work.
  - Speeds up research!
- The ability to easily recover versions of your project in the same folder.
  - Recovering a specific version in a predecessor project code.
- The ability to understand the motivation behind changes via comments.
  - Crucial for continuing existing research projects.
- The ability to increase the reproducibility of scientific results.
  - Basis for cross-linking of data, source code and publications / reports.

- An effective and easy to use software with a set of commands for version control.

Software engineering: version control

Git basics on a single slide

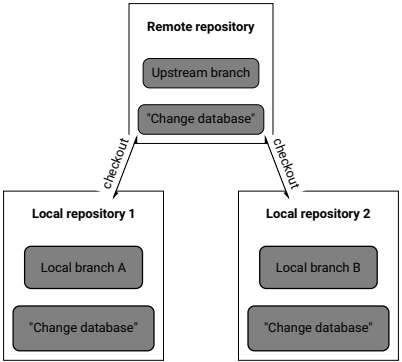
TECHNISCHE UNIVERSITÄT DARMSTADT

- The code/text folder is called a **repository**.
- An online folder shared with the team is the **remote repository** (short: remote).
- Create a new version: **checkout** a new **branch**.
- Integrate with another version: **merge** with a **branch**.
- Add changes in a branch: **add** changes.
- Integrate changes into a branch: **commit** changes.
- Share changes with others: **push** to **upstream repository**.
- Get latest changes: **pull** from the **upstream repository**.

TECHNISCHE UNIVERSITÄT DARMSTADT

Learn basic git *concepts*, they are the same everywhere.

- Git in 15 minutes
- Git within Matlab
- Feature branch workflow
- GitLab for beginners



- **Although git tracks only changes, every repository is still a complete copy of the project.**
- Offline work is supported!

Git conflicts

- A file is changed differently on two branches and a merge is needed.
- Two team members edit the same file at once.

Modularity reduces conflicts and speeds up teamwork

- Book chapters as separate files vs. book chapters as folders and sections as separate files.

Software engineering: version control

Modularity via Separation of Concerns and Single Responsibility

TECHNISCHE UNIVERSITÄT DARMSTADT

- University research teams (like our LEIA lecture team!) are generally small (2 - 5 members).
- ▸ Separation of Concerns (SC) and ▸ Single Responsibility Principle (SRP) significantly simplify the branching model.
- **Separation of Concerns**: code is organized in non-overlapping layers and sections.
- **Single Responsibility**: functions or classes perform single clear tasks.
- SC and SRP can be applied to any software.
- Dogmatism should be avoided: single responsibility vs less responsibilities.

Simple version-control branching model

Separation of Concerns and Single Responsibility

TECHNISCHE UNIVERSITÄT DARMSTADT

- University research teams *working on the same project* are generally small (2 - 5 members).
- Separation of Concerns (SC) and Single Responsibility Principle (SRP) significantly simplify the branching model.
- **Separation of Concerns**: code is organized in non-overlapping layers and sections.
- **Single Responsibility**: functions or classes perform single clear tasks.
- SC and SRP can be applied to any software.
- Dogmatism should be avoided: single responsibility vs less responsibilities.
- OpenFOAM already uses object-oriented and generic software design patterns.

Simple version-control branching model

Change integration

TECHNISCHE UNIVERSITÄT DARMSTADT

Maintainers (postdocs, experienced Ph.D. students) manage the integration.

- Keep the branching model as simple as possible.
- Main and development branches are protected and managed by Maintainers.
- Maintainers are responsible for git tags and cleanup:
  - **Main**: integrations from *accepted publications* and *development branch*.
  - **Development**: integration of *(CI)-tested improvements*.
  - **Feature**: SRP reduces git-conflicts with researchers working on different files.
- Complex branching workflow $\Rightarrow$ complications with onboarding new members.

(Test Driven Development)

Program CSE tests first

TECHNISCHE UNIVERSITÄT DARMSTADT

TDD[8] for CSE

- Define verification and validation tests at the start.
- Focus placed the final result: interpolation, integration, discretization, PDE solution, physics.
- Top-down, instead of bottom-up test coverage.
- Don't go overboard with unit-tests 🎓: extend unit-tests when debugging a failing CSE test.
- Focus kept on tests with real-world (publication) input.

[8] Freeman, Steve, and Nat Pryce. Growing object-oriented software, guided by tests. Pearson Education, 2009.

(Test Driven Development)

Verification and validation tests define the Application Programming Interface

TECHNISCHE UNIVERSITÄT DARMSTADT

- **New code**: it is easier to program the API you wish for, if you are its first user.
  - Make the class interface easy to use correctly and difficult to use incorrectly[9].
  - Reduce number of function arguments, single responsibility, clear naming, ...
- **Legacy code**: extend existing API without modification.
  - OpenFOAM: understanding class hierarchies, *finding a base class with Runtime Type Selection and a virtual function to overload.*
- **The test application is the solver application with a different input.**
  - If possible, testing and solution is done by the same code.
  - This prevents code duplication.
  - Data output and additional checks can be disabled by (compile-time) options.

[9] Scott Meyers. 2014. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 (1st. ed.). O'Reilly Media, Inc.

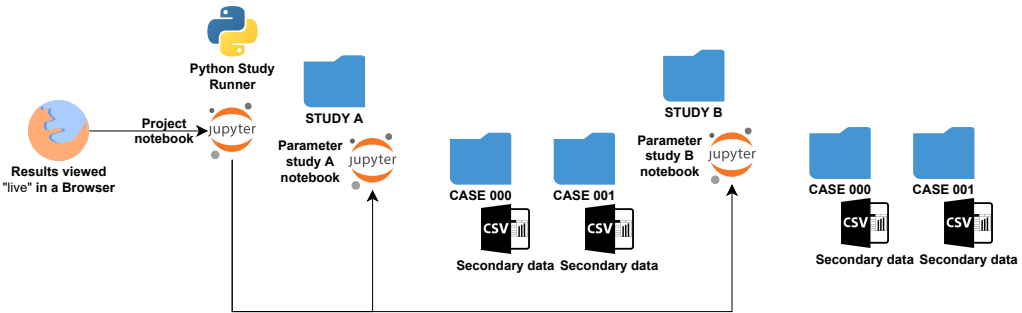
Jupyter notebooks[10]

- **Documentation**: geometry, initial and boundary conditions, error norms, comparison data.
- **Processing**: verification errors (conservation, convergence, stability), validation errors.
- **Result analysis**: very straightforward, interactive, remote.

[10] https://jupyter.org/

Test Driven Development
(Parameter tests)

TECHNISCHE UNIVERSITÄT DARMSTADT



Test Driven Development

Parameter tests: primary data (simulation results) organization

TECHNISCHE UNIVERSITÄT DARMSTADT

- The quality of CSE software is measured using verification and validation data.
- Effective comparison with others (previous versions) hinges on data organization.

- **Legacy code**:
  - use the existing folder structure and parameterization tools 🎓,
  - The mapping (case000) $\rightarrow$ (parameter vector) must be stored (YAML, ...)
- **New code**:
  1. Simple folder and file structure 🎓
  2. HDF5[11] or other open data format.
  3. Alternative to HDF5: **ExDir**[12]

[11] https://www.hdfgroup.org/solutions/hdf5

[12] Dragly, Svenn-Arne, et al. "Experimental Directory Structure (Exdir): An alternative to HDF5 without introducing a new file format." Frontiers in neuroinformatics 12 (2018): 16.

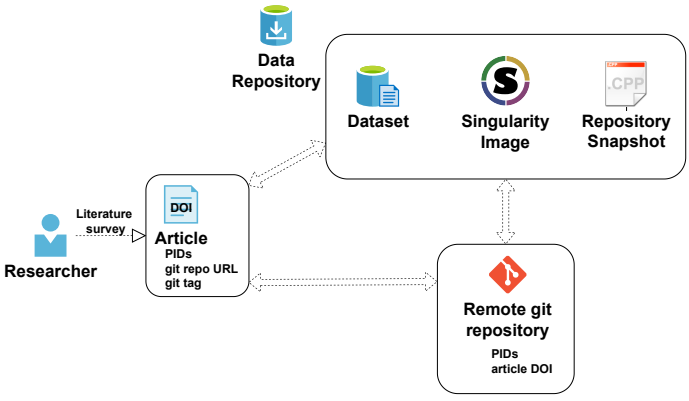
pandas.MultiIndex CSV with metadata for secondary data

- `pandas.MultiIndex` saved in "metadata columns".
- **Metadata is repeated**: not an issue for the small secondary data!
- Metadata in columns $\rightarrow$ `pandas.MultiIndex` $\rightarrow$ strongly simplified data analysis.
- **Direct readable export of tables to LaTex!**

VELOCITY_MODEL	H	L_INF	O(L_INF)	EPSILON_R_EXACT_MAX	O(EPSILON_R_EXACT_MAX)
SHEAR_2D	0.125000	0.032961	1.833407	0.032961	1.833407
SHEAR_2D	0.062500	0.009249	1.955529	0.009249	1.955529
SHEAR_2D	0.031250	0.002385	1.988745	0.002385	1.988745
SHEAR_2D	0.015625	0.000601	1.997178	0.000601	1.997178
SHEAR_2D	0.007813	0.000150	1.999294	0.000150	1.999294
SHEAR_2D	0.003906	0.000038	1.999294	0.000038	1.999294

Cross-linking data, source code and reports/publications

Schematic diagram



(Cross-linking data, source code and reports/publications)

Singularity

TECHNISCHE UNIVERSITÄT DARMSTADT

- Whence the Singularity Image[13]?
  - More intuitive than Docker: **Singularity handles images as files.**
  - Built for HPC from the start.
  - Doesn't require root rights.
  - Results as *actual files*, not "data in spinning containers".
  - Maps user folder to the container: result data remains on the host.
- Why not replace Docker with Singularity within GitLab CI?
  - We're learning how to do this using GitLab custom executors.
  - Does the workflow still survive publish-or-perish 🎓test?
- Why a source-code snapshot on-top of the image and the repository?
  - Repositories get migrated, deleted, and some researchers still fear images.
  - Quick and direct access to source code from the publication.

[13] https://sylabs.io/docs/

(Cross-linking data, source code and reports/publications)

Singularity simplifies reproducibility

TECHNISCHE UNIVERSITÄT DARMSTADT

- The source code and the data stored in the image can be quickly reproduced.
- Article reviewers can clone, build, run and visualize easily.

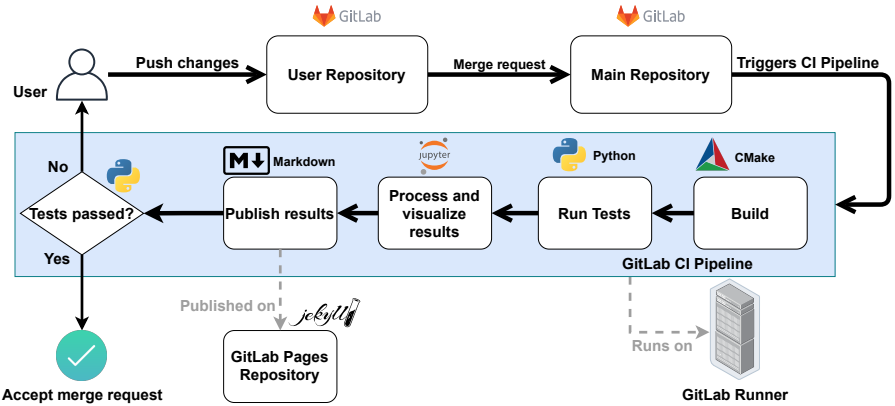
Example: Singularity Image from an active review

- Clone the code repository from the image:
  `geophase-JCOMP-D-19-01329R2.sif clone geophase`
- Build:
  `geophase-JCOMP-D-19-01329R2.sif build geophase build`
- Run tests:
  `geophase-JCOMP-D-19-01329R2.sif run-tests geophase build`
- Open the jupyter notebook:
  `geophase-JCOMP-D-19-01329R2.sif jupyter-notebook geophase`

(Continuous Integration with result visualization)

Schematic diagram

TECHNISCHE UNIVERSITÄT DARMSTADT



TECHNISCHE UNIVERSITÄT DARMSTADT

1. **Short few CPU-core tests**: work-PC 🎓.
2. **Short many-core tests**: obtain a workstation with a 64-Core CPU[14]🎓.
3. **HPC tests**: combine 1. or 2. with an HPC cluster.

An HPC cluster is relevant for production tests and performance measurements.

- This workflow uses coarse ("smoke") tests 🎓
  - Unit tests run for 1. and 2.
  - Convergence ensured for 1. and 2.
  - Is efficient in parallel for 1. and 2.
- **Challenge**: Is it possible to combine 1., 2. and 3. and publish instead of perish 🎓?

[14] Thanks to CRC 1194 at TU Darmstadt.

(Continuous Integration with result visualization)

A GitLab runner with a Docker executor and a local Docker image

TECHNISCHE UNIVERSITÄT DARMSTADT

Build a Docker image for your software, and track the Dockerfile with the project.

Example OpenFOAM Dockerfile on `ubuntu:focal` with "system" open-mpi and scotch.

On the testing machine

- Install Docker and GitLab runner and register the GitLab runner with a Docker executor.
- Configure the GitLab runner in `/etc/gitlab-runner/config.toml` to
  - use a local Docker image, e.g., `image = "openfoam-v2012_ubuntu-focal:latest"`, and
  - never pull images `pull_policy = never`.

TECHNISCHE UNIVERSITÄT DARMSTADT

- Files created within a job are gone when the job ends.
- GitLab uses **job artifacts** to pass on data from one job to the next.
- **Job artifacts only work with files stored in project's sub-folders.**
- Libraries and applications are passed to other jobs as artifacts.
- Artifacts can be downloaded on the GitLab project website.

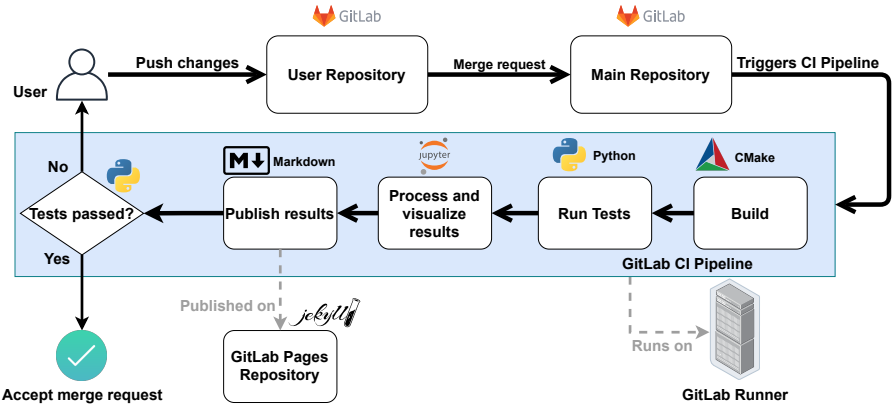
Out-of-source installation: binaries only available outside the repo!

- **Use environment variables to build and pass on artifacts**
- `$FOAM_USER_LIBBIN` folder stores library binaries.
- `$FOAM_USER_APPBIN` folder stores application binaries.
- **Build job**:
  - create artifact folders inside the repo,
  - copy library and application binaries to artifact folders,
  - export artifact folders.
- **Run job: simplified copying of binary artifacts to OpenFOAM folders**
  - `mkdir -p {$FOAM_USER_LIBBIN, $FOAM_USER_APPBIN}`
  - `cp FOAM_USER_LIBBIN/* $FOAM_USER_LIBBIN`
  - `cp FOAM_USER_APPBIN/* $FOAM_USER_APPBIN`
  - Run tests.

(Continuous Integration with result visualization)

Schematic diagram

TECHNISCHE UNIVERSITÄT DARMSTADT



TECHNISCHE UNIVERSITÄT DARMSTADT

```
jupyter nbconvert notebook.ipynb –execute –to FORMAT
```

- Execute each jupyter notebook in the repository.
- Notebooks agglomerate secondary data into `pandas.MultiIndex` CSV files.
- Export secondary data and notebooks in different formats as artifacts.
- **Visualization**
  - Download the artifact and open the notebook 🎓.
  - **Alternative:** publish the notebook as a blog post in a GitLab Static Page project.
  - Notebooks contain information on failing tests.
  - Mapping "caseXYZ" $\rightarrow$ "parameter vector" is crucial for re-starting failed parameter variations!

TECHNISCHE UNIVERSITÄT DARMSTADT

Very straightforward

- Python scripts test secondary data agglomerated by notebooks from simulation results.
- **Examples:**
  - Is the order of convergence of an error norm ≥ 2.0?
  - Is is the difference between simulation and experiment data $\leq 4\%$?

(Continuous Integration with result visualization)
Example

TECHNISCHE UNIVERSITÄT DARMSTADT

Example OpenFOAM CI project

Similarity with other workflows / best practices

TECHNISCHE UNIVERSITÄT DARMSTADT

Our *(subjective)* estimates* of similarity 1 – 5 (higher is more similar), –: aspect not addressed.

DOI	*Branching model*	*TDD*	*Cross-linking*	*CI*	(Meta)data standardization
10.12688/f1000research.11407.1	-	-	-	-	1
10.3934/math.2016.3.261	-	-	-	-	2
10.1371/journal.pbio.1001745	1	2	-	-	-
10.1371/journal.pcbi.1005510	-	-	3	1	3
10.1145/2723872.2723881	1	-	-	1	-
10.1145/3324989.3325719	1	-	-	5	-
10.1371/journal.pone.0230557	1	-	-	1	4
10.1145/3219104.3219147	1	-	-	4	-

**The list may still be incomplete.*

Summary: the minimal workflow

TECHNISCHE UNIVERSITÄT DARMSTADT

Minimal workflow

- Track the status of your project using the GitLab Kanban board.
- Learn and apply the very basics of git, understand the concepts behind branching, merging, commits and working with remote repositories.
- Integrate the changes that worked, don't leave 30 branches open.
- Format the data using established (open-source) formats if possible, if not, at least do this for secondary data (diagrams and tables).
- Share your secondary data on a (TUdatalib) data repository.
- Periodically cross-link research data: publication/report, scripts/code, secondary and primary data.

Acknowledgements

TECHNISCHE UNIVERSITÄT DARMSTADT

Interaction between Transport and Wetting Processes

Funded by the German Research Foundation (DFG) – Project-ID 265191195 – CRC 1194 : Z-INF