RSE Winter of Code Seminar

Robert Speck and **Daniel Ruprecht**
Lehrstuhl Computational Mathematics, Institute of Mathematics
Hamburg University of Technology
In collaboration with
Jülich Supercomputing Centre at Forschungszentrum Jülich GmbH

TL;DR:
What is a really good way to learn new things? With the help of others. In this seminar you will team up with researchers from various domains to see how they write software and help them to get obstacles out of the way. During a talk and a short report you will summarize your findings and tell others about what you did.

Abstract:
How does research software look in practice? How do researchers write and maintain their codes? What obstacles do they face? These questions will be addressed in this seminar.
In this "Winter of Code", participating students will get hands-on experience with cutting-edge research software, giving them a unique glimps of how research software is written, used, maintained, and extended in practice.

Supervised by research teams from the Helmholtz Association, the students will be working on the team's software, a particular application area, and a seminar task from the field of research software engineering. This could range from adding a new feature or repairing a long-standing issue to establishing a new continuous integration workflow, improving test coverage or usability. Active tasks can be found here, participating codes are in the appendix/below: https://github.com/pancetta/RSE_seminar_TUHH_2/issues

The seminar is completed with a 30-minutes presentation of the results and a short, 4-page written report, highlighting the findings.

Prerequisites:
- To get started you will need to know what `git` is and how to work with GitHub (issues). A GitHub account is mandatory.
- Some experience with reading and running other people's code is helpful.
- Detailed requirements (programming languages and paradigms, tools, etc.) are given in the individual issues: https://github.com/pancetta/RSE_seminar_TUHH_2/issues (typically: C++ or Python, but also Fortran)

What you will learn:
- Working in a research software team, good practices for research software
- How other people write and maintain their software
- Talking and writing about research software

Organization

- We will start with an introductory onboarding meeting, where the topic and seminar will be presented.
- Participants get to choose from a of challenges and then start working together with the code owners on these challenges.
- There will be a mid-term meeting to check on the progress of each participant and to identify roadblocks (if any).
- Regular meetings and a close collaboration with the research software teams are mandatory for all projects.
- On-demand meetings are possible if needed.
- In a final, in-person symposium, participants will present their findings in 30-mins talks. This block symposium will happen toward the end of the semester.
- Most of the communication will happen via Zoom.

Assessment

- Your work will be assessed as stated in the module guidelines. You will have to write a report and present your results.
- Criteria are the quality of your report, the quality of your presentation and your level of engagement with the project.

You can find more information at the GitHub page for this seminar:
https://github.com/pancetta/RSE_seminar_TUHH_2

List of participating codes (for a complete, up-to-date list see GitHub Issue):

- **Score-P**: https://www.vi-hps.org/projects/score-p
  Score-P provides insight into massively parallel HPC applications, their communication, synchronization, I/O, and scaling behaviour to pinpoint performance bottlenecks and their causes.

- **PEPC**: http://www.fz-juelich.de/ias/jsc/pepc
  The PEPC project (Pretty Efficient Parallel Coulomb Solver) is a public tree code that has been developed at Jülich Supercomputing Centre since the early 2000s. Our Fortran code is a non-recursive version of the Barnes-Hut algorithm, using a level-by-level approach to both tree construction and traversals.

- **IPPL**: https://ippl-framework.github.io/ippl/
  IPPL is a C++ library to develop performance portable code for fully Eulerian, Lagrangian or hybrid Eulerian-Lagrangian methods. IPPL supports simulations in one to six dimensions, mixed precision, and asynchronous execution in different execution spaces (e.g. CPUs and GPUs).

- **pySDC**: https://parallel-in-time.org/pySDC/
  pySDC is a mathematical Python library for spectral deferred correction (SDC) time-stepping methods and its flavors, esp. the multilevel extension MLSDC and the

parallel-in-time integrator PFASST. It is intended for rapid prototyping and educational purposes.

- **JuPedSim**: https://www.jupedsim.org/stable/
  JuPedSim is a C++ framework designed to support students and researchers in investigating pedestrian dynamics and conducting research related to the development and validation of new models or model features. It enables the analysis of experiments and facilitates the proper visualization of results.

- **PedPy**: https://pedpy.readthedocs.io/stable/
  PedPy is a Python module for pedestrian movement analysis. It implements different measurement methods for density, velocity and flow.