Foundations of Software Engineering

FSE 2023.1

Course Introduction and Organization

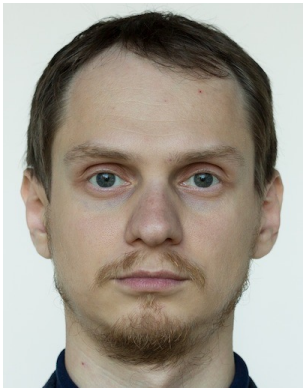
Aleksandr Mikhalev

Fall 2023

Skoltech
Skolkovo Institute of Science and Technology

Your instructor

Aleksandr Mikhalev, PhD:



- 2010-2014, MSU, PhD, defended thesis at INM RAS
- 2014-2015, Skoltech, Software engineer
- 2016-2019, KAUST, ECRC, Postdoc
- 2019-2021, RWTH Aachen, ACoM, Postdoc
- 2022-2023, Skoltech, Senior Research Scientist
- 2023-now, Skoltech, Assistant Professor
- https://faculty.skoltech.ru/people/almikhalev

Motivation of the course

Q: Why learn Software Engineering at Data Science or any other Skoltech Program?

A1: Data science is a very wide field with lots of people working on different aspects. If your new shiny method, that outperforms every other approach in your paper, is hard to download/compile/run/visualize then nobody will actually compare against your approach.

A2: Typical story in science: advisor asks to improve an approach done by other student last year or just to compare it against new SotA. Respect your teammates and save their and your own time by properly developing software.

Motivation of the course

- Most research in DS: programming rather simple ideas
- Most experiments in DS: computational experiments
- Most projects in today's ML: team efforts on software development
- Most projects in today's computational sciences involve HPC and heterogeneous computing, complex numerical libraries
- Most cited papers in ML: papers with great code

It is all about the software!

Motivation of the course

- Many courses evaluate your final project done in a team. Even if you are working on some project along with your scientific supervisor it is a team work. Therefore, your results shall be repeatable in an easy way.
- To use computational resources students must be aware of certain reproducibility tools (e.g., git and docker). Unfortunately, many new Skoltech students lack knowledge of such tools.

It is all about the reproducibility!

Course outline

Goals of this course:

- Provide an introduction into the **ideas** behind software engineering
  - Build automation, version control, scripting, continuous integration, …
- Learn the **tools** commonly used in software engineering
  - Unix, git, docker, vim, make, cmake, …
- Gain the **skills** needed to continue progressing with software development
  - Writing unit tests, building docker images, …

To whom it may concern

This course is just and introduction!

Do take this course if (either applies):

- You have (almost) never used Unix or developed industrial software (e.g. for a living)
- You are expected to work a lot with Unix environments and want to optimize your time by learning how to do things (more) efficiently

Do not take this course if:

- You would like to go deep into details

This course is NOT about

- Programming per se; Algorithms (except for a tiny subset); Project Management; Machine Learning / Big Data (no SWE for model deployment)

Ethics:

- There are multiple better ways to do things
- Unix and SWE is infinite — your instructor is only aware of particular functionality
- Sharing hacks & ideas and contributing improvements is very much welcome!

Practice makes perfect

- This course is very limited in theory and is mostly practice
- SWE is like driving a car or working out: the more you practice, the better you get at it
- Lots of exercises in this course are going to be dumb as hell: this is intentional to make you repeat things many many times

Course timeline

	Term 1B Week 1	Term 1B Week 2	Term 1B Week 3	Term 1B Week 4
Tuesday 9:00 - 12:00	Unix fundamentals: local machine 1	Unix fundamentals: remote machine	Testing software	Project
Thursday 9:00 - 12:00	Unix fundamentals: local machine 2	Building software	Debugging software	Project
Friday 16:00 - 19:00	Version control	Dependencies, reproducibility and docker	Deploying software	Project

Course assessment

The goal of this course is to quickly raise your awareness. To pass the course you need to acquire 75% of the maximal final grade.

The final grade =

40% × Computer labs

40% × Final project

20% × Test/quiz

Course assessment

The goal of this course is to quickly raise your awareness. To **pass** the course you need to acquire 75% of the maximal final grade.

Group assignments (up to 5 students):

- Computer labs (40% of the final grade): copy-paste your terminal commands and outputs.
- Course project (40% of the final grade): implement a simple app using studied development pipeline.

Personal assignments:

- Quiz (20% of the final grade): one hour to solve some easy problems. Enough to try solutions in your terminal.

Course project

- Done in teams of up to 5 students
- IN: Decision what to implement from a scratch.
  Example: calculator app, AI-based image classificatory.
- OUT: a git repository with
  - Different branches and non-linear history in the main branch (e.g., merged paths)
  - Automated build system (e.g., cmake, make). Something must be compiled (e.g, Python exptension module)
  - Testing (e.g., ctest or pytest)
  - Packaged in docker (e.g., an image on docker hub)
  - Tested on different platforms (e.g., Mac or Linux)

Group assessment

Group assignments (up to 5 students):

- Computer labs (40% of the final grade)
- Course project (40% of the final grade)

Split into groups of up to 5 students within Canvas (skoltech.instructure.com) and take your seats together.

Questions?