Course Syllabus

Skoltech

Course Title	Foundations of Software Engineering
Course Title (in Russian)	Основы программной инженерии
Lead Instructor	Mikhalev, Aleksandr

1. Annotation

Course Description

This course is intended to serve as an introduction into basics of everyday industrial software engineering. Oftentimes students seek to obtain proficiency in complicated subjects such as machine learning, algorithms, or computer vision, but lack basic literacy in software engineering and therefore have little practical skills required to carry out research or industrial projects. The course focuses on Unix fundamentals (shell and command line, tools such as vim and awk, scripting, filesystem, streams and pipes, parameter parsing, remote machine and ssh, etc.) and software engineering in teams (code review and version control, building and auto-making programs, reproducibility and containers, testing and test-driven development, improving code style, software deployment and APIs, etc.).

Course Description (in Russian)

Настоящий курс служит введением в технологии промышленной программной инженерии. Как правило, студенты с математическим (в том числе прикладным) и естественно-научным образованием стремятся обрести, прежде всего, высокий уровень компетенций в сложных предметах, таких как машинное обучение, алгоритмы или компьютерное зрение, однако не имеют достаточного уровня знаний или навыков для выполнения исследовательских или промышленных проектов со значительным объемом программно-инженерной работы. Основу курса составляют работа с системой Unix (например, работа в командной строке, написание скриптов, работа с файловой системой, потоками и перенаправление ввода-вывода, организация взаимодействия программы с окружением, работа с удаленной машиной и безопасные оболочки, и др.) и командная программная инженерия (например, анализ кода и системы контроля версий, воспроизводимость и контейнеры, сборка и авто-сборка программ, тестирование и разработка на основе тестирования, рефакторинг и повышение качества кода, развертывание и интерфейсы программирования и доступа к приложениям, и др.).

2. Basic Information

Course Academic Level

Master-level course suitable for PhD students

Number of ECTS credits

3

Course Prerequisites / Recommendations

Basic knowledge of programming in imperative languages (e.g., C, python, or Fortran).

Type of Assessment

Pass/Fail

Mapping from grades to percentage:

Pass: 75

Term

Term 1B (last four weeks)

Students of Which Programs do You Recommend to Consider this Course as an Elective?

Masters Programs	PhD Programs
Advanced Computational Science Data Science	Computational and Data Science and Engineering

Maximum Number of Students

	Maximum Number of Students
Overall:	70
Per Group (for seminars and labs):	70

Course Stream

Science, Technology and Engineering (STE)

3. Course Content

Lecture, lab and seminar hour distribution among topics

Topic	Summary of Topic	Lectures (# of hours)	Seminars (# of hours)	Labs (# of hours)

Topic	Summary of Topic	Lectures (# of hours)	Seminars (# of hours)	Labs (# of hours)
Unix fundamentals: local machine	1) Command line, shell, environment. Alternative shells, dotfiles. Shell completion. File structure and basic file manipulation commands. 2) Data wranging, find, grep and searching for file contents. Executable files and shebang. Shell and python scripting, command-line editing with VIM.	2	2	2
Version control	Version control fundamentals, git, commits, branching and merging, multiple remotes, tagging. Specific repository organization: git flow.	1	1	1
Unix fundamentals: remote machine	Remote shell and secure shell, ssh agent, terminal multiplexing, package control, downloading data (wget and rsync), copying data.	1	1	1
Building software	Compiling programs: gcc, g++, nvcc and alike. The Unix make system and makefiles. CMake and writing portable configuration and building systems.	1	1	1

Topic	Summary of Topic	Lectures (# of hours)	Seminars (# of hours)	Labs (# of hours)
Dependency management, reproducibility, and docker	Fundamentals of containers and the docker CLI, using available docker software. Creating your own docker containers.	1	1	1
Testing software	Fundamentals of software testing. Testing Python applications: unittest, nosetest and friends. Test-driven development. Continuous integration.	1	1	1
Debugging software	Debugging fundamentals. gdb, pdb debugger. Visual debuggers in PyCharm. Profiling software.	1	1	1
Deploying software	Fundamentals of packaging, versioning software. Python modules and packages. __init__.py, PYTHONPATH, sys.path. Deploying python packages. Writing READMEs. deb packaging.	1	1	1

4. Learning Outcomes

Please choose framework for learning outcomes Knowledge-Skill-Experience

Knowledge
Familiarity with fundamentals of Unix operating system
Familiarity with version control systems

Knowledge
Familiarity with testing, test-driven development, and continuous integration
Familiarity with software deployment and packaging
Familiarity with principles of reproducibility

Skill
Practical skill of using the Unix command line, writing shell scripts, working with a local and a remote Unix machine, and working with large-scale data.
Practical skills of version control systems (git) and software versioning.
Practical skills of writing software tests and configuring continuous integration.
Practical skills of packaging python software.
Practical skills of configuring dockerfiles.

Experience
Experience in working with local and remote Unix machines, writing scripts, and working with data on Unix servers.
Experience in distributed software development in teams: versioning software, maintaining codebase, fixing bugs, testing, deploying, etc.

5. Assignments and Grading

Assignment Types

Assignment Type	Assignment Summary	% of Final Course Grade
Computer Labs	Programming exercises on your laptops. Each lecture we start a new programming exercises and discuss solutions. Everything that is not finished becomes your homework.	40
Test/Quiz	Tests in through Canvas with limited time on each topic.	20
Final Project	Final project to show your knowledge and experience with docker, CMake, Makefile and command line tools.	40

6. Assessment Criteria

Select Assignment 1 Type

Computer Labs

Input or Upload Sample of Assigment 1:

Input Sample of Assignment 1

Your colleague has asked you to prepare a dataset for a machine learning project in the form of a train/validation/test split, shuffled randomly over a specified column. Bad news is, due to the security

reasons, the data is only available on a remote machine in colleague's private directory, and he could not recall which specific directory stores that file.
Help your colleague solve the problem.

Specifically, perform the following actions:
— Use a secure shell to connect to a given remote server
— When performing connection, you may want to setup a ssh key-based authentication to speed up connection
Use Unix commands to perform a series of actions:
— Locate a file with a particular name on the home directory of a given user
— Copy the file over to your directory
— Extract the contents of an archive file containing the dataset
— Extract only the specified rows and columns from a file
— Shuffle the file randomly on a specified column
— Split the file unto segments of specified length for train/val/test datasets
— Archive the file into a specified format

As output, provide the shortest possible sequence of your actions and unix commands required to solve the task. Use a numbered list and provide comments where necessary (e.g., what is expected to be printed on the console, what environment variables have been set, etc.). Please also provide any optimizations that you have made or any actions that have made your work faster and/or more comfortable.

Hint: you may check the contents of your bash history while preparing this report.

Assessment Criteria for Assignment 1

PEER REVIEW:
1. Critical criteria (10 points):
— The student is able to connect to a given remote server
— The student is able to properly use Unix commands to:
locate a file with a particular name on the home directory of a given user;
copy files between destinations
extract archives
select rows from tabular files
select columns from tabular files
shuffle text files
split text files into chunks
archive files

2. Additional (bonus) criteria (5 points):
— The student has setup himself an ssh key to optimize server access
— The student has setup himself a handy environment
— The student has not renamed files, but has rather used more elaborate tools to work with complex filenames, encodings, etc.

Select Assignment 2 Type

Test/Quiz

Input or Upload Sample of Assigment 2:

Input Sample of Assignment 2

1. Which of the following statements defines an environment variable in a shellscript?
* echo ${OUTPUT_XYZ_DIR}
* (YES) BINARY=${BIN_DIR}/voronoi_1
* (YES) XYZ_FILES=$(find ${INPUT_XYZ_DIR} -type f -name "*.xyz")
* mkdir -p ${OUTPUT_XYZ_DIR}

2. Which of the following statements assign a program's output into a user-defined environment variable?

* find ${INPUT_XYZ_DIR} -type f
* INPUT_HDF5_FILENAME=/home/artonson/datasets/eccv_test/points/high_0.02/val/val_1024_0.hdf5
* (YES) OUTPUT_HDF5_BASENAME=$(basename ${INPUT_HDF5_FILENAME%.*})
* rm -rf ${INPUT_XYZ_DIR}

Assessment Criteria for Assignment 2

Multiple choice quiz. Correct/incorrect answers are marked.

Select Assignment 3 Type

Final Project

Input or Upload Sample of Assigment 3:

Input Sample of Assignment 3

For the Final Project, your task is to refactor a specific Github repository with multiple dependencies into more portable, reproducible, automatically buildable, better documented, deployable software.

You are given the old legacy code that implements a method that you want to reimplement, and a list of requirements for the reimplementation.

Depending on the requirements, you have to get to know the machine and remote access, build the code, create tests and wrappers for major methods in the code, debug the software, make the library deployable, deploy the code in a container.

Assessment Criteria for Assignment 3

Each project may have its own requirements, but the general evaluation criteria include checking availability of:
- code repository,
- clear implementation structure,
- auto-builds (Makefiles, CMakeLists, etc.),
- tests,
- clear documentation,
- reproducible environment or a container,
- working (debugged) software/code,
- convenience of running the software (entrypoints, data getting scripts, etc.)

Input or Upload Sample of Assigment 4:

Input or Upload Sample of Assigment 5:

Input or Upload Sample of Assigment 6:

Input or Upload Sample of Assigment 7:

Input or Upload Sample of Assigment 8:

Input or Upload Sample of Assigment 9:

In the next question we ask you to define general categories of the course. What does your course teaches in broad terms?

7. Textbooks and Internet Resources

You can request at most two required textbooks. Additionaly, you can suggest up to nine recommended textbooks.

Required Textbooks	ISBN-13 (or ISBN-10)

Required Textbooks	ISBN-13 (or ISBN-10)
Martin, R. C. (2018). Clean architecture: a craftsman's guide to software structure and design. Prentice Hall.	9780134494166
Gamma, E. (1995). Design patterns: elements of reusable object-oriented software. Pearson Education India.	9780201633610

Recommended Textbooks	ISBN-13 (or ISBN-10)
Your UNIX/Linux: The Ultimate Guide. Sumitabha Das	9780073376202
Chacon, S., & Straub, B. (2014). Pro git (p. 456). Springer Nature.	9781484200773
Poulton, N. (2019). Docker Deep Dive. JJNP Consulting Limited.	9781521822807
Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates-Head First Design Patterns -OReilly (2008)	9780596007126
McConnell, S. (2004). Code complete. Pearson Education.	9780735619678
Pilone, D., & Miles, R. (2008). Head first software development. " O'Reilly Media, Inc.".	9780596527358

Web-resources (links)	Description
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow	git flow framework
https://docs.docker.com	docker documentation

8. Facilities

Software
docker, git, and some other standard programs shipped with Linux

Equipment
Unix workstations

9. Additional Notes