Course Title (in English) | Foundations of Software Engineering
---|---
Course Title (in Russian) | Основы программной инженерии
Lead Instructor(s) | Artemov, Alexey

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. In this course, our goal is to bridge the gap between basic programming skills commonly taught during BSc programs and the industrial-grade engineering required by top-notch MSc, PhD, or R&D positions.

Topics include:
- Unix fundamentals (shell and command line, scripting, filesystem, streams and pipes, parameter parsing, remote machine and ssh, etc.),
- Software engineering in teams (code review and version control, reproducibility and containers, testing and test-driven development, improving code style, software deployment and APIs, etc.),
- Software design (team organization, software specifications, software project management, software design methodologies, object-oriented software design, etc.).

As a project, the students will be required to work in teams to design, engineer, test, and deploy a real large software system using the principles described in this course.

Course Prerequisites / Recommendations | Basic knowledge of programming in imperative languages (e.g., C, python, or Fortran).
## 2. Structure and Content

### Course Academic Level

Master-level course suitable for PhD students

### Number of ECTS credits

3

<table>
<thead>
<tr>
<th>Topic</th>
<th>Summary of Topic</th>
<th>Lectures (# of hours)</th>
<th>Seminars (# of hours)</th>
<th>Labs (# of hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix fundamentals: local machine</td>
<td>Command line, shell, environment, and scripting. File structure. Executable files and shebang. Alternative shells, dotfiles, process monitors. Shell completion. An idea of unix sockets. find, grep and searching for file contents, data wranging.</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Unix fundamentals: remote machine</td>
<td>Copying data, terminal multiplexing, package control, remote editing with VIM, ssh agent, downloading data (wget and rsync)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Version control</td>
<td>git flow organization, advanced branching and merging, multiple remotes, tagging. git flow organization, advanced branching and merging, multiple remotes, tagging</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Testing software</td>
<td>Fundamentals of software testing. Test-driven development. What is build</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Deploying software</td>
<td>Fundamentals of packaging. Why packages? Versioning software (what are alpha, beta, dev, stable, pre-release, RC, minor and major, builds, etc.). What are python modules and packages. <strong>init</strong>.py, PYTHONPATH, sys.path. python setup.py and PYPI. READMEs. deb packaging</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Docker and reproducibility</td>
<td>Fundamentals of containers and the docker CLI, using available docker software. Creating your own docker containers</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>IDEs and codestyle</td>
<td>Good codestyle and pylint. Pycharm, Clion and friends. vim, emacs and low-end console analogues.</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>The software development process</td>
<td>Different frameworks for the software development process: Waterfall, Agile, RAD, etc. Team roles in software engineering teams</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Software requirements and design</td>
<td>Documents commonly employed in the development Vision and scope (VS), Software requirements specifications (SRS). Software modelling with UML. Fundamentals of project management.</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Object-oriented software design</td>
<td>Object-oriented programming. Conceptual and technical design. UML and how it can be used for designing software.</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### 3. Assignments
<table>
<thead>
<tr>
<th>Assignment Type</th>
<th>Assignment Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Labs</td>
<td>Work within an unfamiliar Unix environment to manipulate large-scale datasets for machine learning. Use Unix commands to copy the data from the remote machine, split these into folders, extract from compressed files, transform using pipes, etc.</td>
</tr>
<tr>
<td>Computer Labs</td>
<td>Work within an unfamiliar git repository to propose an improvement for a software project. Clone the repository to a local machine, create a separate branch, hack the codes, perform testing, and propose the pull request to the project owner.</td>
</tr>
<tr>
<td>Computer Labs</td>
<td>Write stress-tests for an unfamiliar code to understand where the bugs may come from. Use the python testing frameworks to assemble a fully tested software project.</td>
</tr>
<tr>
<td>Computer Labs</td>
<td>Create your own docker image that may be used for running deep learning projects within python. Create a dockerfile from scratch, configure the necessary environment, and test the software within a running container.</td>
</tr>
<tr>
<td>Computer Labs</td>
<td>Implement and package a large-scale python project using the principles of modular programming. Write a custom distribution script and use it to package the project.</td>
</tr>
<tr>
<td>Case study</td>
<td>Create vision &amp; scope, software requirements specifications, and conceptual and technical design documentation for your team's final project.</td>
</tr>
<tr>
<td>Team Project</td>
<td>Final project on software development in teams: work in a large team to develop a fully functional customized web search engine. The developed software must be a highly portable, automatically buildable, well-documented, continuously testable, deployable web service on a cloud. The software must fulfill the requirements and adhere to the vision document specified by the course instructor.</td>
</tr>
<tr>
<td>Test/Quiz</td>
<td>Unix fundamentals general quiz: shell and command line, scripting, filesystem, streams and pipes, parameter parsing, remote machine and ssh, etc.</td>
</tr>
<tr>
<td>Test/Quiz</td>
<td>Software engineering in teams general quiz: code review and version control, reproducibility and containers, testing and test-driven development, improving code style, software deployment and APIs</td>
</tr>
<tr>
<td>Test/Quiz</td>
<td>Software design general quiz: team organization, software specifications, software project management, software design methodologies, object-oriented software design, etc.</td>
</tr>
</tbody>
</table>

4. Grading

<table>
<thead>
<tr>
<th>Type of Assessment</th>
<th>Grade Structure</th>
<th>Activity Type</th>
<th>Activity weight, %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attendance</td>
<td>Attendance</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Computer Labs</td>
<td>Computer Labs</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>Case Study</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Team Project</td>
<td>Team Project</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Team Feedback</td>
<td>Team Feedback</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Test/Quiz</td>
<td>Test/Quiz</td>
<td>10</td>
</tr>
</tbody>
</table>

Grading Scale

<table>
<thead>
<tr>
<th>Grade</th>
<th>Pass:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>75</td>
</tr>
</tbody>
</table>

Attendance Requirements: Mandatory with Exceptions
## 5. Basic Information

### Maximum Number of Students

<table>
<thead>
<tr>
<th>Overall:</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per Group (for seminars and labs):</td>
<td>40</td>
</tr>
</tbody>
</table>

### Course Stream
- Science, Technology and Engineering (STE)

### Course Term (in context of Academic Year)
- Term 1B (last four weeks)

### Course Delivery Frequency
- Every year

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

<table>
<thead>
<tr>
<th>Masters Programs</th>
<th>PhD Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Science</td>
<td></td>
</tr>
</tbody>
</table>

### Course Tags
- Programming

## 6. Textbooks and Internet Resources

<table>
<thead>
<tr>
<th>Required Textbooks</th>
<th>ISBN-13 (or ISBN-10)</th>
</tr>
</thead>
</table>
## Recommended Textbooks

<table>
<thead>
<tr>
<th>Author/Title</th>
<th>ISBN-13 (or ISBN-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your UNIX/Linux: The Ultimate Guide. Sumitabha Das</td>
<td>978-0073376202</td>
</tr>
<tr>
<td>Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates-Head First Design Patterns -OReilly (2008)</td>
<td>978-0596007126</td>
</tr>
</tbody>
</table>

## Web-resources (links)

<table>
<thead>
<tr>
<th>Link</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow">https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow</a></td>
<td>git flow framework</td>
</tr>
<tr>
<td><a href="https://docs.docker.com">https://docs.docker.com</a></td>
<td>docker documentation</td>
</tr>
</tbody>
</table>

## 7. Facilities

### Equipment

Unix workstations

### Software

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

## Labs for Education

Computer Lab

## 8. Learning Outcomes
## Knowledge
- Familiarity with fundamentals of Unix operating system
- Familiarity with version control systems
- Familiarity with testing, test-driven development, and continuous integration
- Familiarity with software deployment and packaging
- Familiarity with principles of reproducibility
- Familiarity with principles of software design

## 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.
- Practical skills of writing software design documents.

## 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.
- Experience in basic software development process: writing supplementary documentation, software modelling, following a software development framework.

### 9. Assessment Criteria

**Input or Upload Example(s) of Assignment 1:**

| Select Assignment 1 Type | Computer Labs |

**Input Example(s) of Assignment 1 (preferable)**
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

1. Critical criteria (10 points):
   - The student understands how to connect to a given remote server
   - The student understands how to 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.

Input or Upload Example(s) of Assignment 2:
1. Which of the following statements defines an environment variable in a shell script?

* 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.

Input or Upload Example(s) of Assignment 3:

Select Assignment 3 Type

Case Study

Input Example(s) of Assignment 3 (preferable)

Your goal for the Team Project is to develop a modular search engine for a portion of Website X. For the system to function correctly, individual modules must be integrated into the whole system while maintaining its conceptual integrity.

To develop the system, the course project teams need to work together to achieve the common goal. Specifically, they must coordinate using formal documentation and design compatible and verifiable system components.

For the case study, you need to create the following design documents for your component:

— Vision & scope document
— Software requirements specification
— Conceptual design
— Technical design

Assessment Criteria for Assignment 3

PEER REVIEW

Your documentation should be written for your peers to read as internal documentation. During their part in the system design, they will rely on your system’s interfaces and external requirements to formulate the design of a compatible components.

During the peer review process, your goal as a reviewer will be to rate the quality of the design and technical documentation provided by your peers on a variety of factors, such as:

— Does the designed system solve the tasks outlined in the requirements specification documents?
— Does the designed document specify a complete system? Are there design gaps?
— Are the external interface requirements specified in a clear way?
— Is internal structure of the component clearly specified and reasonable?

Input or Upload Example(s) of Assignment 4:
#### Select Assignment 4 Type
Team Feedback

**Input Example(s) of Assignment 4 (preferable)**

Please rate your peers, taking into account their relative productivity, contribution, supportiveness, and other factors, trying to answer questions such as:
- What is their specific contribution to the development of the component?
- What is their specific role to the development of the component?
- Would you recommend them to improve on their technical skills?
- Would you recommend them to improve on their communication skills?
- Would you recommend them to improve on their teamwork skills?

Your feedback will be anonymous and will not be visible to your peers or even your course instructor.

**Assessment Criteria for Assignment 4**

**PEER REVIEW**
Each student is evaluated individually on a 1-10 scale, with aggregated recommendations for further development formed automatically and reported to the student.

**Input or Upload Example(s) of Assignment 5:**

#### Select Assignment 5 Type
Team Project

**Input Example(s) of Assignment 5**

Your objective for the team project is to develop component X to be used for your peers in the context of a larger software system. The resulting system should function according to its requirements documentation but also satisfy many non-functional requirements such as deployability, testability, user-friendliness, portability, extendability, etc.

This assessment’s goal is to evaluate the resulting component based on the mentioned criteria and provide a specific plan for component’s further development and improvement.

**Assessment Criteria for Assignment 5**

**PEER REVIEW**
Each component is evaluated by the users of the component on a variety of criteria corresponding to software quality traits such as:
- Is the component performing well on the task it solves?
- Is the component well-documented?
- Is the component easy to use?
- Is the component safe to use?
- Is the component reliable?

Another evaluation includes open questions such as:
- How can the components’ design be improved?
- How can the components’ performance be improved?
- How can the components’ codestyle be improved?
- How can the components’ deployment be improved?

This feedback is visible to the component team.

An aggregated score is computed by averaging feedback scores from the users of the component.

**10. Additional Notes**

**Free Style Comments (if any)**
Topics of the course may be subject to reordering when teaching. I will think more about which teaching order is optimal during the summer.