

WS12.P12

Software Infrastructures for Teaching at Scale

Editors: Jakob Lykke Andersen & Ulrik Nyman

Contributors: Luís Cruz-Filipe, Michele Albano,
Florian Echtler, Jens Myrup Pedersen,
Miguel Campusano, Aisha Umair, Marco Chiarandini,
Md Saifuddin Khalid & Jiayan Wu

May 1, 2024

Abstract

This report describes the lessons learned through the DIREC project *P12 - Software Infrastructures for Teaching at Scale*. This includes both the lessons learned from actual sub-projects as well as the meta lessons coming from a distributed DIREC project.

Contents

1	Introduction	3
1.1	Context Within DIREC	3
1.2	Existing Cloud Infrastructure for Teaching at Scale	3
1.3	Communication and Commitment	4
1.4	Outline	4
2	Summary of Experiments	4
3	A: Support System for Programming Projects	6
3.1	Teaching Context and Objectives	6
3.2	Software and Infrastructure	6
3.3	Outcome and Recommendations	7
4	B: Teaching Software Engineering at Scale	9
4.1	Teaching Context and Objectives	9
4.2	Software and Infrastructure	9
4.3	Outcome and Recommendations	10
5	C: Automatic assessment of software assignments: Scalable Teaching	11
5.1	Teaching Context and Objectives	11
5.1.1	Case A: Web Technologies	11
5.1.2	Case B: Advanced Object-Oriented Programming	12
5.2	Software and Infrastructure	12

5.3	Outcome and Recommendations	13
5.3.1	Automatic Testing	13
5.3.2	Code Review Quality Assessment	14
5.3.3	Software Engineering Outside Coding Components	14
5.3.4	Difficulty of Use for Students	14
5.3.5	Difficulty of Use for Professors	15
6	D: Distributed Systems Graphical Interface	16
6.1	Teaching Context and Objectives	16
6.2	Software and Infrastructure	16
6.3	Outcome and Recommendations	17
7	E: Enhancing Cybersecurity Education Through Developing for Haaukins	19
7.1	Teaching Context and Objectives	19
7.2	Software and Infrastructure	20
7.3	Outcome and Recommendations	21
8	F: Software Infrastructures for Teaching at Scale: The Case of Introduction to Statistics at Technical University of Denmark	23
9	G: Web Apps for Education Management	24
9.1	Teaching Context and Objectives	24
9.2	Software and Infrastructure	25
9.3	Outcome and Recommendations	25
10	Conclusion and Future Work	27
10.1	Conclusions on Project Organization in General	27
10.2	Conclusions on Organization of this Project	27
10.3	Tools and Infrastructure Development	27
F	Appendix	29

1 Introduction

The introductory part of this report is written by the coordinators of the project, Ulrik Nyman (AAU) and Jakob Lykke Andersen (SDU). The central part of the report, Sections 3 through 8, is contributed by participants in the project, noted in each of their sections. In this central part are reports on experiments with using software to aid in teaching. In Section 2 we provide a very brief overview of the aim of each experiment, and in Section 10 we provide conclusions on the project as a whole.

Outside the central report sections, when using *we* this is referring to Ulrik and Jakob.

1.1 Context Within DIREC

This project, P12, was one of three predefined projects within Work Stream 12 (WS12) of DIREC¹. It should be noted that on the surface it may seem to intersect greatly in scope with one of the other predefined projects, P10, *Learning Technology for Improving Teaching Quality at Scale*. However, in P10 the focus is on how to teach using technology, while we here in P12 focus on the technology itself.

Due to the project being predefined, it did not as such have a true owner with a vision, and the initial task was therefore to develop an actual project with a more clear direction.

The funding structure for the project was predefined such that each of the original partner universities of DIREC were allocated a relatively small amount of funds (approx. 90 kDKK for CBS and 130 kDKK for the other universities). As we can only directly administer funds at our own universities (AAU and SDU), we needed to find proxies at the other universities to unlock the complete project. In the middle of the project, the DIREC management decided to redistribute the WS12 budgets based on which partners were actually willing to participate in this project.

1.2 Existing Cloud Infrastructure for Teaching at Scale

As part of the initial project refinement we explored the possibility of using a common cloud infrastructure for developing software solutions that would be easily shareable among the participating universities. We therefore investigated which infrastructure was already in place, and found multiple systems:

- AAU, AU, and SDU has an ongoing collaboration on UCloud² through DeiC³.
- KU has ERDA⁴.
- DTU has the DTU Computing Center (DCC)⁵.

¹<https://www.direc.dk>

²<https://ucloud.sdu.dk>

³<https://www.deic.dk/>

⁴<https://erda.ku.dk/>

⁵<https://www.hpc.dtu.dk/>

However, these systems are only available to their respective universities. It was difficult to obtain information about how these systems have been used for teaching, and we only learning about a few instances. Though note, generally the systems are aimed at research purposes.

For teaching at scale it can become increasingly important with integration into Learning Management Systems (LMSs). Our initial investigation revealed that also in this area there is little alignment between universities, and at least four different systems are in use (Moodle, Brightspace, itslearning, Absalon), thus making development of a common software suite more complicated.

1.3 Communication and Commitment

Due to the highly distributed nature of the funding structure, an early goal was to find a representative from each partner university, to be responsible for local distribution of funds, promotion of the project, and in general act as contact person. However, this had limited success, due to a variety of reasons: it was during the early part of the project where the scope was still being defined, each representative had very limited funds and thus less incentive and decision power, and some representatives were rightfully skeptical of burdening colleagues with tasks. A general issue we encountered was the basic task of getting information to teachers about the existence of the project. As we only had access to our own departments we had to rely on our representatives for forwarding information. Even with successful forwarding, the messages can be easy to dismiss as spam-like, especially due to the rather low amount of funds available. From meetings in Work Stream 12, this is not a challenge isolated to this project.

1.4 Outline

The main part of this report consists of descriptions of experiments related to the project (Sections 3–9). Most experiments were supported with funds from the project, though the contribution by Florian Echtler, Section 4, is based on previous experience.

In Section 2 we provide a brief summary of the experiments, and in the end in Section 10 we conclude with recommendations for future work.

2 Summary of Experiments

The details of each sub-project are described in the following sections. In this Section we briefly summarize the type of experiments being conducted.

Three of the experiments aimed directly for providing (semi)automated feedback on programming tasks for students in Computer Science and Software Engineering courses (see Sections 3, 4, and 5). The automation was realized using a combination of Git for version control, various continuous integration (CI) tools for executing tests, and some amount of custom software to manage the setup. For example, the experiment described in Section 5 includes software for teachers to give feedback and score assignments on aspects that can not be covered by full automation. In all experiments the tests are designed by teachers, which implies a relatively large amount of work to set up a course the first time.

The experiments described in Section 6 and 7 aimed at extending existing software platforms for learning, with the latter focusing on the core task of creating more exercises and the former on enhancing the platform itself with additional features.

Section 8 describes an experiment focusing on another aspect of running courses. Here there were already existing courses using multiple software platforms, but the issue was with onboarding of new teachers and teaching assistants to such courses. The aim of the sub-project was to develop standardized and unified guidelines for this onboarding.

The experiment described in Section 9 regards the development of two tools used for planning of respectively exams and student project assignment. Here the aim was to provide graphical interfaces for two existing tools, one for assigning students to groups based on preferences, and one for scheduling exams such that students have a more even workload in an exam period.

3 A: Support System for Programming Projects

This project focused on developing support tools for programming projects, and was coordinated by Luís Cruz-Filipe at University of Southern Denmark (SDU). The software was developed by a student helper (Thomas Wulff Heissel), paid for by the project.

3.1 Teaching Context and Objectives

The Bachelor programs in Computer Science and Mathematics at SDU include a mandatory course on Introduction to Programming in their first semester (course codes respectively DM574 and DM536).

One of the goals of the course is to teach the students to follow a *contract* — developing a program that respects an interface that was previously agreed upon, so that independent teams can work on different parts of the same project simultaneously, and expect these parts to connect seamlessly. This is achieved through a multi-stage group project, where the group develops three parts of the same program independently, following a contract that is given to them.

To give them the possibility to test whether they are following the right contract, the students should be able to test their development in the context of the whole program.

3.2 Software and Infrastructure

The goal of this project was to develop a simple online platform to run a program consisting of the students' code together with a correct implementation of the remaining components of the project. This platform should be easily configurable, so that it can later be used in other courses without requiring major changes. The starting point is a testing system previously used in another course, which is not configurable and can only be extended by substantial reengineering of the source code.

The platform we developed includes a correct solution of the whole project (nine Python files). The students can access the platform through a browser, where after authentication they are given the chance to upload a ZIP file. Afterwards, a container is created containing a copy of the correct solution; any files uploaded by the students whose names match those in the project overwrite the ones previously there. The project is then run on some predefined tests, and the output is shown in a browser window. This may be correct output, a transcript of any error messages generated while executing the program, or a timeout message from the interface (if the program did not terminate after two minutes).

All the information about the structure of the expected input and the format of the tests to execute is written in configuration files specific for this project, which can be edited by system administrators (including teachers).

The backend records some information for monitoring purposes, including the details of each submission (usernames of the group members, timestamp, a copy of the uploaded files, and the generated output). This information can be visualized by administrators in aggregated form.

3.3 Outcome and Recommendations

With respect to the original objectives, the project was a complete success. The interface was up and running by the time the students started working on the project. There were only very minor adjustments that needed to be made, which also allowed the student helper to spend some time analyzing the logs, propose and implement some small improvements.

An evaluation of the students' performance revealed two issues that require attention in future editions of the course:

1. Some groups used the system as a testing interface, instead of designing their own tests.
2. Some groups performed an absurdly high number of tests, sometimes re-submitting their code every few minutes during a couple of hours.
3. The testing interface is extremely slow, due to the heavy use of containers, resulting in an unexpectedly high number of timeouts even for correct solutions.⁶

Regarding the first point, the course objectives and the project description both clearly state that students need to argue for the correctness of the software they write — either through documented self-explanatory code, in the case of very simple functions, or by designing relevant tests. Due to the availability of the test interface, several groups simply assumed that getting a positive result from their submission meant that their program was working correctly, despite:

- the output from the test interface clearly showing that the program was *not* giving correct results;
- the lecturer mentioning several times in the lectures that the test interface was exclusively meant to check that they were following the correct contract (in terms of function names and input/output types).

Regarding the third point, the timeout was increased to 3 minutes, and the students were informed that timeouts were not necessarily indicative of problems in their solution — as long as the output generated by their program before the timeout was meaningful.

Based on these observations, we plan to implement the following changes in future editions of the course.

1. Restrict the number of submissions to make the test interface a scarce resource. The current plan is to limit to one submission per group per day.
2. Change the output text to make it clear that “success” does not mean that the program is correct, but only that it syntactically matches what is expected.

If more resources are available, there are some additional changes that we would like to consider.

⁶The official solution ran in around 3 seconds on the teacher's computer. The timeout given to the students was 2 minutes.

3. Currently, the interface is not interactive, meaning that the program runs without interference and the users only receive the final output from fixed tests. This limits the kind of projects that the students can be given. We would like to expand the interface so that the students can give input during execution.
4. The current platform relies heavily on containers, which was a design choice of the legacy system due to security requirements from SDU. It would be nice to investigate whether an alternative implementation can be found that is less resource-intensive, in the interest of making the system both faster and more environmentally-friendly (wrt. energy consumption and heat/CO₂ generation).

4 B: Teaching Software Engineering at Scale

In this section, we briefly describe a course conducted by Florian Echtler from 2018 to 2020 while still teaching at Bauhaus-Universität Weimar in Germany.

4.1 Teaching Context and Objectives

The course in question was simply called “Software Engineering”, and had been taught before to an audience of 15–20 undergraduate Computer Science students. However, after a restructuring of the educations, this course was now intended for a different master study programme called “Digital Engineering”. The audience now consisted of students who had already finished an undergraduate engineering degree (mostly construction and electrical engineering), with little to no prior programming knowledge. This led to the following challenges:

- Heavily increased course size, to about 50 students.
- Little/unevenly distributed prior programming knowledge

The end goal was still to give the students foundational knowledge of software engineering, however, the practical part of the course needed to be adapted to meet the challenges mentioned above.

4.2 Software and Infrastructure

Our initial step was to rebuild the exercises along a topic that might be more relatable for the engineering students, which was building a raytracer. This approach has several advantages:

- Easily produces visual output with a minimum of external dependencies, writing a file is sufficient.
- Straightforward to write test cases for verifying correct operation, e.g., by comparing the output value for certain pixels or vectors.
- Sufficiently extensible to illustrate multiple important software engineering concepts such as polymorphism and GoF (Gang of Four) design patterns⁷, e.g., the visitor pattern.

To avoid having to manually extract and check every code submission, we used [Github Classroom](#) instead of Moodle as a submission platform. This gave us the opportunity to a) introduce students to a modern development workflow using version control, b) provide them with a starting template including test cases and build files, and c) use continuous integration (CI) tools for checking the submitted code automatically against the test cases.

For the CI tooling, we opted to use [Travis](#), which offered a free tier suitable for our purpose and a seamless Github integration that would run the test cases automatically after every commit. Since the course was last conducted in 2020, Github has introduced their own CI system with Github Actions which offer the same functionality and would likely be preferable today, due to closer integration.

⁷https://en.wikipedia.org/wiki/Design_Patterns

4.3 Outcome and Recommendations

One unexpected challenge we faced during the first iteration of the course was heavy plagiarism of the exercise assignments. For subsequent iterations, we tried to address this issue by a) making the course repositories private, and b) using plagiarism detection software. We initially used [MOSS](#), which worked with varying degrees of success, and later switched to [JPlag](#), which proved to be slightly more reliable. For example, unlike MOSS, JPlag is able to exclude template code from the comparison, which would otherwise be flagged as plagiarized.

Overall, there was a significant amount of setup work required for the initial course iteration. While the workload for actually checking the assignments was noticeably lower, the initial work for creating test cases and example solutions that matched the test cases was extensive. Consequently, this approach works best if it can be reused several times, with the same assignments (and possibly slightly modified test cases, if the plagiarism issue persists across semesters).

The relevant lecture slides for this course are available as open educational resource at <https://github.com/floe/software-engineering>, with a video lecture recording on [YouTube](#). Both resources are licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The exercises, code examples, and test cases are also available on request.

5 C: Automatic assessment of software assignments: Scalable Teaching

Scalable Teaching is a web application used by the SDU Software Engineering professors to help them assess students' software assignments. It has been mainly used by Miguel Campusano (Assistant Professor) and Aisha Umair (Teaching Associate Professor) during their courses.

5.1 Teaching Context and Objectives

The students from SDU Software Engineering learn several programming techniques and designs that need to be assessed by their professors. However, the large amount of students during the bachelor (around 100 students per course) makes the assessment extremely challenging. Professors are expected to not only grade students' software assignments but also provide software engineering quality assessment to their source code, as it is a critical component of their learning and education.

In particular, professors face the following two challenges for both the correctness and quality of students' solutions: 1) effective assessment of assignments, exercises, etc., 2) providing quick feedback to students.

5.1.1 Case A: Web Technologies

In the first iteration of the Web Technologies course, Scalable Teaching was used to grade students' software assignments automatically. To allow this, the tool implemented the Continuous Integration / Continuous Delivery (CI/CD) technique, which automatically tests software when people from the software team incorporate new features into the software.

Scalable Teaching uses CI/CD to automatically run tests when students implement the features asked for in a particular software assignment. Then, if every test passes, the assignment is considered complete. If not, the assignment is considered a miss.

While this worked well, it did not provide granular feedback to the students. It could have given feedback in an effective way on which part of the software assignment failed or why. Moreover, a pass-or-fail assessment can be too hard for specific assignments. Finally, the tool only provides feedback on the correctness of the assignment but not on the quality of the source code.

In the next iteration of the Web Technologies course, Scalable Teaching was updated to support 1) 100%-scale grading and 2) source code feedback. The tool supported 100%-scale grading by dividing large tests into several small tests, which are assigned a certain percentage of the total amount. A 100% assessments mean that the assignment passed every test.

Source code feedback was implemented using the concepts behind source code review, where a developer assesses the quality of the source code written by another developer. Instead of doing this automatically, Scalable Teaching provides a user interface that allows students to assess the quality of their fellow students, effectively implementing peer review in the context of software assignments. This strategy aims to 1) give students feedback on the quality of their code, and 2) introduce students to the concept of software development peer review.

5.1.2 Case B: Advanced Object-Oriented Programming

This course requires educators to grade students' code assignments using a point-based grading system instead of binary (pass/fail) grading. Manual point-based grading of programming tasks is, however, tedious and challenging. Having to grade many students is not only cumbersome, but it can often become hard to keep track of everything.

To address this, Scalable Teaching is used to grade the students' assignments more efficiently. This tool allows educators to define a rubric, where each evaluation criterion is associated with a specific number of points. The tool collaborates with a VS code extension to directly combine grading criteria with the student files, making it easier for educators to quickly review each rubric evaluation criterion and ensure the student fulfills the requirements. Thus making it harder to make mistakes. To minimize bias, the educator/grader is unaware of who they are grading. Moreover, the student can see who graded his assignment, the earned points, and the feedback.

5.2 Software and Infrastructure

Scalable Teaching offers several features designed to provide efficient and effective assessment/feedback of students' assignments using Continuous Delivery / Continuous Integration and source code review. The feedback can be provided through comments, explanations/suggestions, or highlighting errors. In addition to the feedback, it also provides a means to track and analyze students' progress in various course activities. It can accommodate a large number of students and scales well as the number of students grows.

Scalable Teaching is a ready-to-use web application set up in an SDU server. Nevertheless, the source code is open, allowing anyone to set up Scalable Teaching on their servers. However, people setting up the tool should be aware they must also set up the CI/CD server independently to allow automatic assessment.

This tool uses a component-based/modular approach, dividing the overall system functionality into independent, reusable components/modules. The modules can be combined in various configurations to achieve the desired behavior of the system. To this end, the administrator/professor creates a course and then creates tasks within a course. This is followed by attaching modules to the tasks in order to configure them. The implemented modules are listed below:

Mark as Done. This module allows students to mark their tasks as *Done* upon completion, typically used for exercises, allowing them to track their progress through a semester. The students do not need to handle anything, nor does the professor check if they completed their tasks.

Link Repository. Some tasks are simple assignments that are listed as a set of instructions for the students to follow. However, some more technical assignments and exercises require using a predefined codebase or other libraries to be available to the students. The link repository feature allows the professor to use a repository from a GitLab server as a reference that students can consult. This feature is significant when students do not need a new repository for each task but simply need a codebase they can work on. This module is also a precursor for other modules; specifically, the *Template* module requires enabling this module.

Template. Once enabled, instead of just receiving access to a codebase, the linked repository now functions as a *starting point* for the students. Upon starting the assignment, students will receive a forked version of the linked repository, allowing them to work on their solution. The last commit to the codebase before the deadline functions as the final hand-in for the student.

Preload Repositories. This module works similarly to the previous one but creates repositories for students ahead of time to avoid strain on the GitLab server.

Build Tracking and Automatic Grading. This module allows the professor to watch when students build assignments and provides valuable feedback about when students are active during an assignment. As students push code, the information about each build is logged. The build tasks module can also be used to monitor if a student has passed the assignment in a pass/fail manner. This is done by creating specific tests that run during build steps to ensure students fulfill the assignment’s goals.

Subtasks and Automatic Grading. A binary pass/fail grade is not always good enough as it fails to provide much nuance for the professor. Subtasks allow the professor to create small tasks that are part of the assignment, where each subtask can be assigned a point value. The subtasks can be used with automatic grading where build steps can be “linked” to subtasks, allowing the professor to inspect which parts of the assignment they have completed.

Manual Grading and Feedback. Enables professors and TAs to manually inspect student files in the code viewer, giving them a brief view where each subtask can be easily marked and commented on. This feature can effectively distribute the tasks among selected teaching assistants (TA) to distribute the workload equally. Student names can also be anonymized, such that the professor or TA is never faced with the student’s name — both the student and professor are given a pseudonym that allows students to inquire about their grading by providing this pseudonym. Lastly, this module also enables peer feedback, enabling students to comment on their peers’ (anonymized) code.

Project Files. This module protects files from unauthorized editing (by the students). Specifically, some files are used to indicate what tests should run. If the students change these, they could cheat the system by getting the build to pass that would ordinarily fail.

Automatic Download. This module enables automatic download of the students’ repositories after the task deadline. Assignments are kept for two years, after which they are automatically deleted from the server.

5.3 Outcome and Recommendations

In this section, we group some of the outcomes, challenges, and recommendations we encountered when building and using Scalable Teaching in the context of software engineering education.

5.3.1 Automatic Testing

The iterative approach of building a tool allows us to fix some of the early limitations of the same tool. For example, we initially developed a binary automatic grading system, limiting grading possibilities and feedback. Then, we implemented a 100%-scale grading system, which can be automatic or manual.

Nevertheless, we noticed that using automatic testing in the context of education can be pretty different than in a typical software development context. Tests can not only provide correctness but also lead to solutions, limiting the student's range of possible solutions to just a few (or sometimes only one). When using automatic testing, it is key to design and build proper tests to assess correctly what the professor needs to know from the student.

5.3.2 Code Review Quality Assessment

Scalable Teaching provides code review quality assessment based on a peer review system, where students review their classmates. To avoid useless or harmful reviews, professors and teaching assistants can moderate the system and give feedback on the code and the reviews. While this has worked well in the past, and students appreciate the feedback, more research must be done to confirm if this kind of review is helpful for students. Moreover, this review is being designed as a post-assessment phase after the assignments are graded. For other types of assignments, it may be helpful to include code quality assessment as part of the grading process

5.3.3 Software Engineering Outside Coding Components

A key component of software engineering is to design valuable software for a stakeholder, according to their requirements. After the design is done, the software can be implemented. Scalable Teaching is not designed to grade or provide feedback on the design component.

Usually, requirements and design decisions are handled using a report (i.e. a PDF document). An interesting future work avenue for Scalable Teaching will be how to handle these documents, especially if they are joined with an already written piece of software. Professors usually review documents and source code simultaneously, and Scalable Teaching can be extended to support this type of evaluation.

5.3.4 Difficulty of Use for Students

Students are expected to learn how to manage source code during their education. Therefore, they should learn to use a code repository tool (e.g. GitLab), automatic testing, continuous integration/delivery, programming languages, frameworks, APIs, etc. These techniques and tools are essential for students to learn. Still, they distract students from the fundamental complexity of tasks, especially for students early in their education.

To focus students' assignments, we can use interactive code execution directly in Scalable Teaching. This will allow students to work immediately on their exercises without installing systems or worrying about configuration issues by working directly on the web user interface instead of on their computers. With this solution, students can focus on the fundamental complexity of the assignment (solving the problem itself) instead of using time on the incidental complexity (anything outside the problem itself, e.g., installation problems).

5.3.5 Difficulty of Use for Professors

While the platform itself is working, the module system and writing tests can be complex for professors. Instead of using a module system, Scalable Teaching should present several *pre-load configurations* for assignments, where professors will select from what they need. For example, instead of loading *Link Repository*, *Template*, *Build Tracking* and *Automatic Grading*, they should be able to pick an automatically graded assignment from a specific GitLab repository.

Moreover, professors still need extra documentation to use the tool effectively. Documentation can be provided by user manuals, videos, talks, etc., showcasing different features of Scalable Teaching.

6 D: Distributed Systems Graphical Interface

This project aimed at the creation of a graphical user interface for an existing teaching tool. The tool is an emulation platform for teaching distributed systems at Aalborg University (AAU). The project also involved unit tests and error handling and detection.

This project was initially led by Peter Gjør Jensen, who submitted the request for funding and selected the student to be employed, namely Thomas Møller Jensen. Later, Peter stepped down since he decided to teach other courses, and his co-chair Michele Albano took the lead on this project.

6.1 Teaching Context and Objectives

An important learning goal for problem-based learning (PBL) universities is related to experiential learning, and in this context it was considered that the course Distributed Systems required a platform to emulate the systems described in the lectures. The course is taught at the 7th semester of the Computer Science and Software educations, and recently also on the 7th semester of the CS-IT and Data Science educations, all at Aalborg University.

Initially, the platform was developed by Peter and Michele, and by the Teaching Assistants of the course. However, the platform was not user friendly according to the students who used it, since it had no visualization of the emulated systems, and it provided limited and non-intuitive feedback to the users.

With the goal of facilitating the learning process of the students, this project aimed at fulfilling the following goals:

- adding a GUI to make the exercises more interactive,
- adding unit tests (for student self-assessment), and
- improve on error-handling and error-detection (e.g. infinite loops).

The project was carried out in 2022, during which Thomas was the lead developer. During fall 2022, the improved platform was used in the course Distributed Systems, both by the lecturer to demonstrate parts of the systems, and by the students for experiential learning.

6.2 Software and Infrastructure

The code of the improved emulation platform is available as open source on Github at <https://github.com/DEIS-ools/DistributedExercisesAAU>. The platform is modular and allows two operation modes:

- a *synchronous* mode (also called *stepping* mode in the documentation), where the execution is articulated into rounds. In each round, (i) all emulated systems receive incoming messages, then (ii) all systems execute their algorithm, then (iii) all systems send out messages if required by their algorithm;
- an *asynchronous* mode, where the systems are scheduled independently from each other. The only synchronization happens when a system waits for an incoming message.

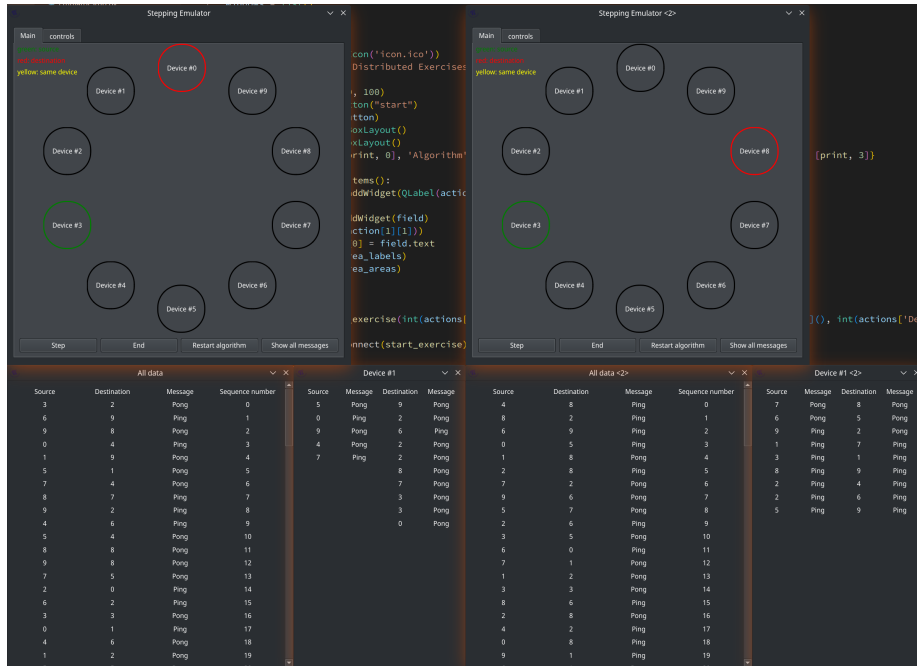


Figure 1: Graphical user interface developed in the project.

The platform consists of a number of Python files, and documentation. In particular, the main folder contains

- a **README** that acts as documentation for the platform, and describes the exercises for the Distributed Systems course;
- **emulators** folder, which contains the engine for both the synchronous and asynchronous execution modes;
- **exercise_runner.py**, which is run by the user to execute the emulator via the command line;
- **exercise_runner_overlay.py**, which executes the GUI developed in this project (see for example Figure 1);
- **exercises** folder, which contains the skeletons for the exercises, named **exercise_1.py** for the first lecture, up to **exercise_12.py** for the last one. These are the only files that should be modified by the students.

6.3 Outcome and Recommendations

After a first development of the system, the project was run in unison with the execution of the Distributed Systems course in Fall 2022, to collect the feedback of the students to steer the development effort. We can argue that the project is a step forward with the digital transformations of teaching, and it received positive feedback from the students. Still, during the project execution

a number of malfunctions were found by the students and had to be corrected, which on the other hand acted as a quality assurance process for the platform.

The outcome of the project is usable in future Distributed Systems courses. Still, a number of directions for improvement can be proposed.

First of all, the usability of the platform could be improved by adding inline documentation of the GUI, and better error messages when the student develops wrong code.

Moreover, unit tests for the exercises could be used not only to find out the most common mistakes, but also to verify that the solution provided by the student is correct. This approach would be inspired by best practices such as test-driven development.

Finally, as the topics taught in the Distributed Systems course change over time, the platform will require to evolve to cover the new topics with new exercises.

7 E: Enhancing Cybersecurity Education Through Developing for Haaukins

This project was conducted at the Department of Electronic Systems at Aalborg University. In collaboration with Andreas Knudsen Alstrup and under the guidance of Jens Myrup Pedersen, this part of the report explores the impactful endeavor of enhancing cybersecurity education through the development efforts for Haaukins⁸.

7.1 Teaching Context and Objectives

This section highlights a significant initiative aimed at contributing to cybersecurity education through development for Haaukins. The project involved the creation of challenges customized for Haaukins, a dynamic cybersecurity platform designed to provide hands-on training and competitive experiences. These challenges are hacking exercises that encompassed diverse domains including reverse engineering, binary exploitation, cryptography, forensics, web security, and miscellaneous fields, thereby offering a comprehensive training landscape.

The roots of this initiative lie in addressing a critical challenge within cybersecurity education — the scarcity of skilled cybersecurity professionals. In response, the endeavor aimed to spark interest in cybersecurity among young individuals by delivering engaging and relevant learning opportunities. The challenges developed for Haaukins were designed to align with the educational goals of cybersecurity, equipping students with practical skills that mirror the intricacies of real-world cybersecurity scenarios.

Moreover, these challenges aimed to bridge the gap between theoretical knowledge and practical application. By actively engaging with cybersecurity concepts, students were encouraged to cultivate critical and innovative thinking while tackling complex issues with the challenges developed. The wide array of challenge categories not only enhanced participants' technical prowess but also nurtured a hacker's mindset, enabling them to discern and mitigate security vulnerabilities effectively.

The project presented a series of challenges encompassing various domains, each designed to test and enhance participants' skills in different areas of cybersecurity and penetration testing. Some examples of challenges are:

- In the realm of web exploitation:
 - Challenge 1: Participants focused on bypassing authentication through simple SQL injection (SQLi), highlighting the importance of secure coding practices.
 - Challenge 2: This exercise was created, a simulated e-commerce environment using Vue.js and Golang. It aimed to provide engaging web exploitation challenges with a user-friendly experience and robust backend architecture.
 - Challenge 3: This challenge required exploiting the search functionality to gain unauthorized access to user information using the SQL UNION operator.

⁸<https://www.cybertraining.dk/haaukins/#/>

- Challenge 4: A challenge in development, where participants hid injected code as a product review to steal user authentication tokens.
- In the forensics domain:
 - Challenge 5: Participants tackled steganography, uncovering hidden data within an image and successive layers of data. The challenge involved handling corrupted files, Base encodings, and Morse code within a sound file.
 - Challenge 6: A Modbus challenge revolved around Russia’s power grid. Players exploited the Modbus protocol to set power plants to zero, observing deactivation and energy production decrement.
 - Challenge 7: Participants analyzed vulnerable router firmware compiled using OpenWRT, seeking a concealed folder using Linux tools.
- In the miscellaneous category:
 - Challenge 8: Participants learned to hack an IP camera using the RTSP protocol and the Cameradar tool. Accessing camera credentials, they connected to the stream containing modified content with the challenge flag.

These challenges collectively aimed to enhance participants’ cybersecurity skills and deepen their understanding of various attack vectors and vulnerabilities across web exploitation, forensics, and miscellaneous domains. The diverse range of challenges provided participants with opportunities to learn and apply techniques critical to securing digital systems and networks.

The impact of this initiative resonated through its integration into various courses and teaching activities, especially within education programs such as Cyber and Computer Technologies, as well as the Master’s in Cyber Security. Additionally, its influence extended to community-based activities related to Cyber Skills and De Danske Cybermesterskaber, where the platform and its challenges found practical application.

The subsequent sections delve into the intricacies of Haaukins’ software and infrastructure, the outcomes of these collaborative efforts, and recommendations for future utilization. This collective endeavor plays a pivotal role in nurturing the landscape of cybersecurity education, empowering individuals with hands-on experiences that transcend theoretical concepts and enrich the collective cybersecurity expertise.

7.2 Software and Infrastructure

The architecture and infrastructure of Haaukins represent a cornerstone of its effectiveness in enhancing cybersecurity education. At its core, Haaukins serves as a highly accessible and automated virtualization platform, catering to a wide spectrum of users ranging from students to teaching staff. The platform’s architecture is designed to create a seamless and user-friendly experience, minimizing barriers to entry and maximizing engagement.

Each virtual lab within Haaukins is composed of a Kali Linux machine and a series of challenges, often implemented as Docker containers. These challenges

span various cybersecurity domains, covering a range of topics from reverse engineering to cryptography. Participants engage with these challenges within isolated environments, facilitating safe and controlled learning experiences. Access to the Kali Linux machine is facilitated through Apache Guacamole, enabling seamless and secure browser-based interaction, thereby eliminating the need for complex setups on participants' end.

Automation plays a pivotal role in Haaukins' operation. Setting up an event or class becomes a streamlined process, where teachers or organizers can select challenges and set parameters through a user-friendly web interface. This triggers the automatic provisioning of labs and event websites, streamlining administrative tasks and ensuring a consistent experience for participants. The automation also extends to user registration, making it simple for participants to self-register and be assigned to a specific lab.

The synergy between software and infrastructure is central to Haaukins' success. Challenges, being implemented as Docker containers, are lightweight and portable, ensuring efficient resource utilization. This flexibility allows Haaukins to operate across distributed servers, providing scalability to accommodate concurrent events with large participant numbers. Currently distributed across servers at various institutions, the platform is in the process of reworking its architecture to allow for even more flexible distribution based on available resources.

In conclusion, Haaukins' software and infrastructure embody a fusion of accessibility, automation, and scalability. The platform's user-centric design and automated processes enable educators to seamlessly offer diverse cybersecurity challenges while fostering engaging and secure learning experiences for participants. The ongoing evolution of Haaukins' architecture further underlines its commitment to continuous improvement, providing a robust foundation for the future of cybersecurity education.

7.3 Outcome and Recommendations

This endeavor successfully developed valuable challenges for Haaukins, enriching the learning experience on the platform. These challenges, a result of this project, have notably contributed to cybersecurity education.

In the initial stages of development, the process of onboarding and configuring Haaukins challenges, along with effectively integrating cybersecurity learning goals, posed obstacles for the developer. However, through close cooperation with the Haaukins team, these hurdles were swiftly overcome. The collaborative efforts and expertise shared ensured a seamless transition, allowing the challenges to align cohesively with the educational objectives, thus enhancing the overall learning experience for participants.

Covering various domains, these challenges engaged students with real-world scenarios, fostering critical problem-solving skills. Seamlessly integrated into Haaukins, they facilitated active participation and effective learning. Contributing to the Haaukins' challenges library of more than 500 challenges, the exercises developed hold substantial educational value, enhancing the platform's depth. Aligned with the project's goal of enriching learning opportunities, they successfully addressed specific aspects of cybersecurity education.

Looking ahead, this momentum will bolster Haaukins' role in cybersecurity education. Commitment to scalability, usability, and diverse content will sustain

the platform's pivotal educational role.

Future endeavors entail continued challenge development, aimed at encompassing the latest Cyber Security knowledge. This ongoing dedication aims to equip cybersecurity professionals with necessary skills to tackle evolving digital challenges. As the platform advances, it signifies contributors' instrumental role in cybersecurity education advancement.

8 F: Software Infrastructures for Teaching at Scale: The Case of Introduction to Statistics at Technical University of Denmark

The work by Md Saifuddin Khalid and Jiayan Wu is a larger sub-project, and their report is included in [Appendix F](#).

9 G: Web Apps for Education Management

This project aimed at developing two web interfaces for previously developed tools for student-project assignment and exam scheduling. The purpose of this was to make the tools more broadly available and more easily usable. The project participants were Marco Chiarandini, Associate Professor at the Department of Mathematics and Computer Science of the University of Southern Denmark (SDU) and Andreas Twisttmann Askholm, student assistant. The project ran from July 10, 2023 to October 15, 2023 and has been financed with 75,000 DKK. All budget has been used to pay the salary of the student assistant, Andreas.

9.1 Teaching Context and Objectives

The two software tools developed in the project are used to help teachers at SDU in managing their time while maximizing student satisfaction. They address the following challenges in the organization of teaching activities, which are common in the context of higher education:

- Assigning students to project topics and supervisors. The challenge is finding an outcome that satisfies both students and advisors. In a free process, where students make private agreements with advisors efficiency is lost. The most popular advisors become overloaded, first movers get better options, early agreements can be undone or are regretted. In large settings a free process is simply not viable.
- Scheduling the dates of oral and written exams. The challenge is finding schedules that maximize the distance between exams for each student while also including constraints on the teacher presence and the room availability.

Concretely, at SDU we face the student-project assignment challenge in the following activities:

- BADM500, Bachelor project in Computer Science, approx. 40 students for 20 project topics.
- K04, Internship in the Master in Psychology, approx. 150 students for 70 internship topics.
- FF501, First year project the Faculty of Science of SDU, approx. 240 students for 100 project topics.

Further, we face the exam scheduling challenge in the following case:

- Every semester at the Faculty of Science of SDU, approximately 200 exams for 2400 students to be scheduled in 20 available days.

The student-project assignment tool developed helps in managing the distribution of students to advisors or supervisors in project work. It streamlines the process by collecting in a first phase the descriptions of the offered projects with a desired size of the group of students working on them and capacity limits. In a second phase it collects the student preferences towards the projects. In a third

phase an algorithm finds an assignment maximizing students' satisfaction while taking into account capacity limits of the advisors. Most importantly, it brings fairness in the treatment of students and balance in the work load distribution of the advisors. The assignment is published together with statistics about it and the possibility for the administrators to compare alternatives due to small changes in the offered capacities of the advisors.

The exam scheduling tool collects in a first phase the data about the exams to schedule, the required durations and the restrictions imposed by teacher availability and student registrations to the exams. The data collected contain often errors that must be corrected or invalidated. In a second phase an examination schedule is computed and outputted.

9.2 Software and Infrastructure

The project focused on the development of a front-end consisting of a web interface. The back-end are two existing Python modules that implement the algorithmic solution to the combinatorial problems that are at the core of the tools. The web interface developed in this project are written in Python and Django. They use an SQLite data base and they are deployed in SDU virtual servers with the continuous development tools Git and Jenkins.

The Python module for the algorithmic solution of the student-project assignment is described in this publication:

Marco Chiarandini, Rolf Fagerberg, and Stefano Gualandi. Handling preferences in student-project allocation. *Annals of Operations Research*, 275(1):39–78, 2019.

The Python module for the algorithmic solution of the examination scheduling is unpublished. Both of the modules implement mathematical programming models of the problems, that are then solved by an external mixed integer programming solver. They have been extended to interact with the front end.

Account management for the web application has been handled via Django. It supports both SDU Single Sign On via Microsoft Authenticator and local accounts. The latter is meant for testing purposes and are kept hidden from users. We exposed facilities for editing the database. However, we left the management of user accounts to the Django administration interface. Hence, at the current stage the creation of new accounts can only be done by the main administrators.

The tools are publicly available at:

- Adsigno: <https://adsigno.sdu.dk/>
- Examino: <https://exam.imada.sdu.dk/>

The arrival page asks for the credentials to authenticate the user. Therefore, the content is only available to registered users. Users will then see different content depending on their role which can be: student, teacher or administrator.

9.3 Outcome and Recommendations

It has been a challenge writing these two applications in the short time frame available and with only the basic level of expertise of a student assistant. Above

all it has been challenging testing the application from the point of view of each user role. After the end of the project, development work was still left and was taken up by the project leader. In spite of this, we succeeded in using the system Adsigno in the two editions of the courses K04 and BADM500. The road is open now for the future use of this system.

In the future, we wish to improve the submission of project descriptions which currently can happen either via spreadsheets or via an online form. Above all, currently there is little control to the format of such descriptions and it is not possible to include pictures. Another unfinished feature is the automatic generation of a catalog of project topics. Finally, from the algorithmic point of view, it would be interesting adding preferences also from the side of the advisors towards the students.

At the time of writing the system Examino needs further development and testing. It will be used in the scheduling of exams for the new semester (Spring 2024) but probably only for the data collection phase.

Overall the project delivered one finished web app and one that necessitates further development. The development and the inclusion of several tools and technologies resulted more challenging than foreseen. The complexity of systems built to interact with real users can never be overestimated.

10 Conclusion and Future Work

The conclusion of the project is divided into three parts concluding on three different aspects, described in the following three subsections.

10.1 Conclusions on Project Organization in General

These lessons are intended as advice to a potential DIREC 2.0. The lessons learned from the initiation and scope setting of this project are:

- Ensure clear and motivated ownership of each DIREC project. This project was predefined in the initial DIREC proposal without any clear plan for who should actually carry out the project.
- Collaboration across universities is difficult. Everyone is busy and must prioritize their time. Convincing people to spend time on a project with minimal funds, is very difficult.
- Project managers should be in control of the entire budget of their project. Distributing the budget across multiple universities requires distributed control of the funds, which naturally hinders progress.

10.2 Conclusions on Organization of this Project

These are the lessons learned on the general setup of common software infrastructure for teaching at scale.

- The universities participating in DIREC use many different technologies for supporting teaching, e.g., learning management systems (LMSs) and cloud infrastructure. This can make reuse of software across universities difficult.
- Freedom of use of technologies and infrastructure is different between universities. Some universities have policies for what software can be used, which teachers must follow, while at other universities the individual teacher has almost complete freedom to experiment with new software.
- Despite the declared intention of collaborating within DIREC, the willingness seems in practice to be highly influenced by local policies, resource constraints, and available project funds.
- There are many teachers that are interested and willing to experiment with new technologies, but have resource constraints that make it difficult to carry out experiments.

10.3 Tools and Infrastructure Development

To really support the ease of teaching at scale, tools have to be developed for the given infrastructure that can interact with the surrounding software.

- With the limited funds that each of the sub-projects have been allotted we have still seen a significant amount of progress that has helped the teaching effort in those course.

- There is quite a lot of interest in automated feedback mechanisms, both in developing new setups and in enhancing existing setups. A continuation of this project could concern a consolidation of effort, to not duplicate systems, but accelerate the development of fewer, reusable systems.
- As we see from the Haaukins project in Section 7, even already successful projects that are being used, needs continuous funds to develop new content.
- As we can see from the sub-project at DTU (Section 8), even when tools are available there is still a considerable amount of effort left in implementing them in an organization. Both in regards of on-boarding new teachers, TAs, and ensuring the IT support for the tools.
- From the project by Marco Chirandini (Section 9), we see that teaching at scale also requires tools at higher level than individual courses, e.g., for planning and resource allocation.

If a centralized effort with common tools is a desired goal for DIREC going forward, then we recommend that a lot more resources are to be put into the effort, and political power at higher levels than individual teachers has to be behind it. However, we do not think that this currently is the right approach.

Instead we suggest that tools and approaches are funded in a grass-roots manner, similarly to what effectively happened in this project, to such an extent that they can become stable tools. This could both be tools explored in this report, but also other yet undiscovered initiatives.

F Appendix

The text starts on the next page as it is a directly included PDF.

Software Infrastructures for Teaching at Scale: The Case of Introduction to Statistics at Technical University of Denmark

Project# DIREC P12

Md Saifuddin Khalid & Jiayan Wu

Centre for Digital Learning Technology (LearnT)

Department of Applied Mathematics and Computer Science, Technical University of Denmark

Project Report

Project Co-Funded by

Digital Research Centre Denmark (DIREC)

<https://direc.dk/master-training-network-ws12/>

Table of Contents

1. Introduction	4
2. Context and Methods	4
3. Analysis and Results	5
3.1 Journey of course instructors through software channels	5
3.2 Journey of the coordinators of Teaching Assistants through software channels.....	8
3.3 The five teaching-related tasks and the use of various software systems	11
3.3.1 Communication.....	11
3.3.2 Lecture	11
3.3.3 Exercise	11
3.3.4 Projects	12
3.3.5 Final Exam	12
4. Discussion.....	14
4.1 Piazza for Discussion: Pilot Study in Spring 2023	14
4.2 EdStem for Discussion: Pilot Study in Fall 2023	15
4.3 Brightspace LMS (DTU Learn) for Discussion	16
5. Conclusion	19
Acknowledgement	19
Appendix I	19
Interview Guide for interviewing course instructor and coordinator	19
Appendix II	20
Instructions for direction of Introstat projects	20
General.....	20
Correction instructions	20
Entry sheet	20
Set up navigation bar on LEARN	21
TA Hiring.....	21
Preparation for the course.....	21
A simple check list with todo for each week	21
On streaming.....	23
Feedback to students.....	23
Projects	23
Update the project status list	24
Developing project.....	25
Weeks for deadlines (works for fall, but usually needs to be modified for 02323 in spring)	25
Previous semester dates	26

Administration of projects in Introstat	27
Exam.....	27
Set up of exam	27
On the exam day	29
Calculating the grades after the exam	29
Appendix III	30
Links to Video guide	30

1. Introduction

This project report informs software infrastructures for teaching two introduction to statistics courses at the technical university of Denmark, offered by the department of computer science and applied mathematics. The two statistics courses include various software-based services and custom-built applications, which are used collaboratively by the teaching roles. The software environments, referred to as channel henceforth, include teaching content, formative and summative evaluation, and the information communication among the lecturers, teaching assistants, exam administration, and student support services. In this report, the channels refer to the software infrastructure and the term channel is adopted from the service innovation discipline. Furthermore, teaching at scale refers to teaching large courses comprising 100+ students and *teaching* includes all activities by the teaching roles, that is, not limited to lecture and exercise facilitation. The courses are defined as large courses (100+ students in each course) and given particular attention.

One of the central problems for the teaching team of large courses is onboarding new teaching roles and teaching assistants in the team for teaching introduction to statistics as there are too many software systems including the expected ability to understand, operate and code using multiple systems and languages. While the students' feedback and overall evaluation of the courses are consistently acceptable, the new members of the teaching team and teaching assistants expressed a lack of clear overview and difficulty in learning the process. So, this project contributes by providing visual journey map and descriptive guidelines for onboarding new teaching roles responsible for teaching introduction to statistics and the software infrastructures adopted for teaching or creating content at scale.

The broader objective of the project was to experiment with new software infrastructure and toward that goal Piazza was adopted for the purpose of reducing students' emails reaching the inbox of one course instructor. In Fall 2023, EdStem platform was being tested as an alternative to Piazza as EdStem can also support running R, Python and other codes in one as part of the editor on the platform. The website, shared resources among teaching roles, and other resources have been moved to DTU Compute's GitLab. Furthermore, during summer 2023, it has been decided that the introduction to statistics course will be offered with Python as the problem-solving language instead of R. It has been decided by the boards of studies that the foundation programming language will be Python, which will be used in foundation mathematics, programming, statistics, and software development courses simultaneously. Therefore, from Autumn 2024, the previously developed scripts for generating book, exam questions, assessment rubrics, generating exam results etc. must be changed along with development Python-based statistics textbook according to the existing curriculum. Despite the changes, the journey maps and description of activities in this report is expected to remain as a useful resource for the course instructors, teaching assistants, coordinator of teaching assistants, and study administration.

2. Context and Methods

This project is part of the education stream of DIREC and project number 12. The DIREC P12 project team agreed on the following broader questions for all the partners. The context of this project are the two courses offered by the department of applied mathematics and computer science, technical university of Denmark (DTU).

1. In which course (or other teaching activity) was the experiment conducted? – Introduction to Statistics (02323 & 02402)
2. What was the initial purpose? E.g., which learning goals or resource challenges were you trying to address? – to reduce difficulty in onboarding new instructors, TAs, and consistently following

the diverse software/tools involved errors and rework. The alignment and coordination in the teaching team cause issues in the learning, information communication and interaction process.

3. An elaboration on the challenges/problems/limitations of the setup. What was the result in relation to the initial purpose?
 - 1. Dilemma: One platform that consolidates all desired services, reducing rework and streamlining coordination, is currently unavailable. On the contrary, managing the diversity of platforms (channels), timelines, and coordination for the teaching team is a complex task.
 - 2. There are too many not-so-user-friendly documents that course instructors need to read, remember, and coordinate as part of onboarding and teaching the course.
4. Recommendations for future use of this type of setup. What would the next step be? if any steps are feasible.
 - A journey map or mere mapping the tools (as channels) from the perspectives of teachers, TAs, and students along with the timeline for the interaction through different channels can be made and scheduling notification based on the journey map with support from a secretarial role are expected to improve the experience of teaching, learning and coordination. The lack of resources (man-hours) requires the teaching roles to take up many administrative coordination tasks.
5. A short description of the software and infrastructure setup. This report presents the description of the various inter-connected software used.

For the semi-structured interviews, the interview participants of this project include course instructors and the coordinators of teaching assistant (TA). In this project, 3 course instructors and 2 coordinators of teaching assistants involved in “Introduction to Statistics” were interviewed.

Within this project, concomitant observations were garnered during interviews with participants. Throughout these interviews, participants elucidated their utilization of software and IT infrastructure intrinsic to teaching activities, elucidating the intricacies of their workflows. This observational approach facilitated the discernment of challenges and user pain points embedded within the landscape of teaching activities.

3. Analysis and Results

For teaching the introduction to statistics (02323 and 02402) courses there are four roles involved at DTU Compute. The analysis and results section include (1) the journey maps of course instructors and the coordinators of the teaching assistant (as the role shared tasks of a course instructor/coordinator), (2) five main teaching-related tasks involving various software systems.

3.1 Journey of course instructors through software channels

Only two roles, namely, course instructors and the coordinators of teaching assistant, and their journeys are analysed for mapping the software infrastructure (referred to as channels). Based on the interviews and observations on the use of the different tools, Table 1 summarizes and presents a pseudo journey map of the course instructors. The table 1 summarizes the pains and gains for defining the scopes of improvement.

Table 1. Journey of Course Instructors through software channels

Activities & Channels: Software and Infrastructure	Description of Activities	Gain (+) and Pain Points (-)
Course catalogue: https://kurser.dtu.dk/	The course catalogue is updated by the course coordinator in coordination with the study secretary and submitted for approval by the study board.	- The system allows showing only one language but the 02323 is offered in English in Fall and Danish in Spring. 02402 is offered in Danish in Fall and English in Spring. The students get confused.
Course Website on GitLab https://lab.compute.dtu.dk/users/sign_in https://02323.compute.dtu.dk/ https://02402.compute.dtu.dk/	1. Modify content on website by editing the files under <i>pages</i> , which are loaded as the content and links; update materials for the students (exam schedule, Slides, R codes, textbooks, other materials). 2. Upload PDF versions of PowerPoints formatted content generated using Latex-coded file compilation and upload using SCP through DOS-command lines (in Windows). 3. Solution files of exams are uploaded and linked right after the exam duration ends.	- Annual academic calendar including the non-teaching weeks need to be checked to <i>manually</i> update the dates. -The lecture and exercise venues along with other deadlines need to be updated <i>manually</i> before every semester starts.
Subversion (SVN): Sharing Scripts, Slides, Exams, Guides	1. The teaching team stores and shares all course content by organizing those in an SVN Folder. All the latex scripts for generating lecture content, exam questions, book, exercise questions and solutions are shared through the folder. SVN folder contains instruction guides and notes, exam folder for scripts and codes shared for collaboratively generating the exam files (R files, R markdown files). Teachers manage the files on GitLab in the SVN folder. (Bookkeeping)	+ Teaching team share content through subversion. - Lacks version control: users must always update before committing new files and sometimes creates confusion.
Brightspace Learning management system. https://learn.inside.dtu.dk/	Students Hand-in Assignment, feedback, communication, files, information, quiz, Teachers edit on the admin page. Communication between students and teachers. Announcements by teachers for annotated slides, information on FAQs etc.	-Difficult to make grade or quiz on Learn. -The UI needs to be more user-friendly, there is a gap of generations. - Students send too many emails regarding their mistakes during submission or requesting extension.
Panopto Video Recording and	In both Danish and English, recorded class lectures and	+ Students can access the video lectures.

Sharing: Podcasts/Recorded Videos	activities are edited as short-length videos and structured as topics.	- the number of students participating in the class have reduced significantly. - Some of the videos are not accessible and were not tested after migrating from the previous repository.
Project Rubric creation, calculation etc.	1.Course instructor use R scripts (update on SVN) to create rubric as excel files. 2.Course instructor use Google form to collect feedback from TAs, then the coordinator or course instructor can know their preferences. 3. Course instructor assign projects based on TAs' preferences.	- Lack of a whole picture of assessment. - Need to create a new file every year <i>manually</i> .
Subversion (SVN)	Checking TAs hours on SVN.	- Needs to run the script <i>manually every start of the semester</i> .
TA Hiring:	1.Use R scripts on SVN to find out good students/Previous TA, send out emails to recruit them. 2.There is a folder in SVN, course instructor can find the names of TAs in the last year. 3. When TAs get recruited, their name will be forwarded to a new group.	- Need to do it every semester
TA onboarding	Course instructors create a correction guideline for TAs on SharePoint.	
Discussion Forum Piazza or Learn DTU https://learn.inside.dtu.dk/	Teacher answer students' questions on Piazza or Learn DTU.	Students Ask questions, see questions from previous questions, anonymous. Students can search information. - Questions fresh too fast, some questions are hard to keep track of.
Projects: Submission	Brightspace Learning Management System (LMS): learn.dtu.dk. The assignment feature of the LMS is used. For each of the two project submissions, students can choose from four different topics.	Students submit assignments and TAs provide feedback in bulk
Projects: Assessment & Feedback	1. From the LMS, all the submissions under the four different topics are downloaded and assigned to 13-15 teaching assistants (TAs) by using Microsoft SharePoint.	
Classroom Engagement,	Kahoot & Socrative are used for engaging students during the class.	+ Gamified engagement of students during the class

Kahoot & Socrative		
Exam assessment and Grading Subversion (SVN)	1.Use R Script, Sweve, and Latex to perform exam grading. 2.Use email to send out the results to students. 3.Export exam responses from Digital Exam & List of Participants.	
Q&A System: Piazza, EdStem & Discussion/Messa ge via LMS	Pilot: Piazza in Spring 2023. Pilot: EdStem in Fall 2023 due to the option for compiling code in the message box.	+ One Q/A space for all introstat (02323 & 02402) students. + One of the members from the teaching team can respond to all students in both introstat courses. - Some students still continue to send emails about projects/exercises.
Book/e-Book	Update based on students' feedback/comments and compile the R script for generating the pdf version of the e-book. Send the e-book updated version to the bookstore for printing copies for students https://www.polyteknisk.dk/	- The periodic update of the book is an additional task that needs follow-up by the team. - Issues reported by students regarding the book cannot be reported in one environment for keeping track of. - The updated chapters need to be generated separately and updated on both websites.
Area9 Rhapsode Adaptive Learning Platform	The platform's course environment is created as an additional resource for learning. The accounts are created by importing the list of registered students from https://deltagerlister.ait.dtu.dk/deltagerlister.asp and students are sent email notification by the course coordinator with URL for activating/enrolling in the platform. Periodically, members of the team look at the issues reported by the students and update the content and activities on the platform.	- not mandatory and rate of use/completion is less than 5% - time required to create account, send activation link, and troubleshoot is an overhead administration for the course coordinator/instructor + some students spend quality time and send feedback for correction.

3.2 Journey of the coordinators of Teaching Assistants through software channels

As coordinators of teaching assistants, one or two PhD students from the department support the teaching team by coordinating the team of teaching assistants by allocating two mandatory assignment reports submitted by the students. In the following table, the journey of these project or assignment assessment coordinators containing the software infrastructure, activities, and pain points are presented.

Table 2. Journey of coordinators of Teaching Assistants through software channels

Activities & Channels: Software and Infrastructure	Description of Activities	Gain and Pain Points
Project Assessment, calculation etc. R Script, Sweve, Latex and email	<p>Coordinators of teaching assistant refer to the correction guideline for TAs.</p> <p>The process is as follows:</p> <ol style="list-style-type: none"> 1. The students upload their projects to one of four different assignments on DTU Learn (one for each project type) 2. The coordinator download all the project files from each of the four assignments and save them in four different folders (again one for each type). 3. The coordinator runs a script that summarizes how many projects of each type have been handed in. 4. The coordinator manually distributes the (amount of) projects onto the different TAs taking into account the individual TA's wishes. TAs have prior to this been asked how many projects they would like to correct and which types of projects they would prefer to correct. 5. The coordinator input the numbers into the R script (from point 3). 6. The R script produces correction sheets for the different TAs and zip files containing all the PDFs of the projects that the TA needs to correct. 7. The coordinator upload to SharePoint where the TAs can enter points for the individual questions in the project. 	<p>-Need to do it four times a semester (two times for each course, including distributing reports and processing results), coordinate and distribute the projects based on TA's number and preferences.</p>
Discussion Forum	TAs answer students' questions on Piazza or Learn	<p>+Students Ask questions, see questions from previous questions, anonymous</p> <p>+Students can search for information.</p> <p>-Piazza is hard to keep track of students' questions and it increases TA's workload.</p>
Projects: Submission	Brightspace Learning Management System (LMS): learn.dtu.dk. The assignment feature of the LMS is used. For each of the two project submissions, students can choose from four different topics.	<p>Students submit assignments and TAs provide feedback in bulk.</p> <p>TA can choose to download all the files from SharePoint in a zip-file that the coordinator</p>

		<p>has produced, or they can choose to download them themselves from DTU Learn.</p> <p>Sometimes they just annotate the projects in DTU Learn and will never have to download the files.</p> <p>- Excel tables are very poor, and it is difficult to use, because the table is very large, the content of the cells is poorly readable.</p>
Projects: Assessment	<p>From the LMS, all the submissions under the four different topics are downloaded and assigned to 13-15 teaching assistants (TAs) by using Microsoft SharePoint. TA can correct them locally or on Learn.</p> <p>The process is as follows:</p> <ol style="list-style-type: none"> 1. Download all the reports in bulk, According to the total number of project files, all projects are evenly assigned to all TAs, 2. Send out tasks to different TAs by email. 3. TA refers to the correction sheet on SharePoint 4. TA corrects the reports and gives feedback (They can do it on LEARN directly or make a local PDF correction file and then upload it to LEARN) 	<p>TAs upload corrections (points for each answer in the project assignment) to SharePoint in the files (excel) that the coordinator created for them.</p> <p>TAs give feedback on DTU Learn either as overall comments, as annotations directly in the files in DTU Learn, or as annotations done locally on their own computer and then uploaded to DTU Learn.</p> <p>- Excel tables are very poor, and it is difficult to use, because the table is very large, the content of the cells is poorly readable.</p> <p>-The scripts being old and written to / for R using linux, there are often problems occurring when trying to run the scripts on multiple platforms</p>

		-No plagiarism check, it would be nice with an objective check.
--	--	---

3.3 The five teaching-related tasks and the use of various software systems

The course activities can be broadly categorized into five: 1) Communication 2) Lecture 3) Exercise, 4) Projects 5) Final Exam. For the different course activities, various software infrastructures and procedures are adopted. Based on the interviews, observations, and experience, the activities are reported followed by software/channel-focused journey maps.

3.3.1 Communication

The communications between course instructors and students mainly take place on DTU learn (Brightspace LMS) using the announcement. Piazza and EdStem platforms were used during Spring 2023 and Fall 2023 respectively as part of DIREC P10 project.

Students' inquiries, internal collaboration among the teaching team members and the teaching assistants are communicated via emails. In most cases, students initiate communication and course instructors respond to their questions. Piazza was adopted to reduce the students' emails as there was no discussion forum post on the LMS and students used to email instead. EdStem platform pilot (Fall 2023) was conducted as the platform included most frequently used programming code compilation (including R and Python).

3.3.2 Lecture

Before the semester, all lecture content including the slides and handouts are recompiled and uploaded by the lecturer responsible for the semester. Students are informed that the slides might be updated just before a lecture starts or right after the lecture ends. Kahoot used during the lectures are shared among the teaching roles.

Course instructors need to manually update course information (including exam schedules, links to textbooks and slides, and deadlines, etc.) through Gitlab before the start of each semester. They also need to update and manage course-related files in SVN. Additionally, the shared files used in the course (including scripts and operation guides, etc.) are also in the SVN folder.

3.3.3 Exercise

Exercises are usually carried out in the form of practice sessions. The course instructors use email to notify and assign coordinators of teaching assistants to specific classrooms to provide help to students and answer questions.

To support the exercise, Area9's Rhapsode platform access is given to the students during the beginning of the semester. The adaptive learning platform allows students to exercise in self-, but students report undesired repeated activities like a loop, but the percentage is very low in the first two weeks of the course, the course instructor creates access based on the enrolled student list¹ and sends announcement containing a URL for getting access to the system. One issue is that the students who join later in the course, usually send email to the course instructor and access for each student are required be created separately. The coordination task becomes time consuming for the course instructors and if students leave the course, then the course instructor needs to identify the students

¹ <https://deltagerlister.ait.dtu.dk/deltagerlister.asp>

who dropped out the course then need to renew the list of students. Usually, the students retain access and are not removed from the system,

A custom-built quiz environment complements the exercises, but it does not keep any record of students' access to and interaction on the platform.

3.3.4 Projects

Students are required to submit two mandatory project reports, including the option for re-submission if not passed/approved in the first attempt, and submitted and feedback provided through the learning management system's "assignment" feature. To reduce overhead administration, one PhD student plays the role of a coordinator of teaching assistants and responsible for allocating the submitted projects and coordination of assessment.

The procedure is as follows.

1. The students upload their projects to one of four different assignments on DTU Learn (one for each project type)
2. The coordinator of teaching assistants (TA) download all the project files from each of the four assignments and save them in four different folders (again one for each type).
3. The coordinator runs a script that summarizes how many projects of each type have been handed in.
4. The coordinator manually distributes the (amount of) projects onto the different TAs taking into account the individual TA's wishes. TAs have prior to this been asked how many projects they would like to correct and which types of projects they would prefer to correct.
5. The coordinator input the numbers into the R script in SharePoint. (From point 3).
6. The R script produces *correction sheets* as Excel template for the different TAs and zip files containing all the PDFs of the projects that the TA needs to correct.
7. TAs can download the project zip file on Learn or read the PDF file on learn directly. If the zip contains code, TAs will download it and run it on local computer. (R Script on SharePoint was used to edit contribution).
8. TAs refer to the correction guide on SharePoint, upload the correction file and comment to Learn, or they can leave comments on PDF file on Learn.
9. The coordinator upload to SharePoint where the TAs can enter points for the individual questions in the project.

3.3.5 Final Exam

Activities for the final exam

For the final exam, the process includes the following activities for the IntroStat team and the procedure involving the digital systems are documented subsequently. For the final exam, the activities of IntroStat team are as follows:

1. IntroStat members agree on exam question contribution role, each of the members prepare exam questions, and submit in the shared space.
2. The coordinator checks the submission according to agreed-up deadlines, sends early reminders, and adjusts tasks as appropriate. The process includes preparing the English version of the questions, a correction round by another IntroStat team member, and the final question generation by the coordinator final check and approval internally by the team. Then a Danish version of the questions is prepared by the team members for their respective questions in English and then native Danish language speaker in the team proofreads.

3. Coordinator sends the English and Danish versions of the questions including the solutions to the questions to the censor for review and approval,
4. The coordinator (in collaboration with course instructors, if required) set up online MCQ exam system & set up a test MCQ exam using a previous year's question so that students are prepared for the online environment for the exam,
5. During the exam, the coordinator and one of the course instructors answer students' queries at the exam halls in collaboration with the exam office.
6. The exam coordinator uploads the exam questions with solutions on the course websites and send announcement through the learning management system that the question has been uploaded.
7. Exam coordinator export students' scores for the online exam system,
8. The exam coordinator analyses the students' scores,
9. Exam coordinator handle errors or other factors regarding the exam question (if any),
10. The exam coordinator identifies the cut-off points for the grading according to the Danish 7-point scale,
11. Exam coordinators send email to Introstat team for internal approval regarding grading cutoff points,
12. Email external censor (and CC Introstat team) for getting approval about the grading cut-off points,
13. Submit grades of students using the script-generated content
14. Exam coordinator and course instructors collaboratively send announcements to the courses that the grades have been submitted and remind about the date and procedure for re-exam.

Type and point system of the final exam

The final exam is a multiple-choice 30-question exam, each containing five options, awarded five points for correct answer, and penalized one point for incorrect answers. The exam questions are contributed by an introduction to statistics teaching team, typically comprised of five assistant or associate professors, who plan and coordinate the task using a shared Excel file in their shared SVN repository. An example of the exam plan is shown below:

	Antal spørgsmål	2023dec	Korrektur
Probability		5 PBAC	PBAC
One- and two sample		5 JKMO	JKMO
Simulation based		5 NALO	PBAC
Regression		5 PBAC	ANDBA
ANOVA (One-Way: 2 Spm; Two-Way: 3 spm)		5 NSL	NALO
Count data		5 NALO	ANDBA
Kap. 1 opgaver tilføjes af tovholder			
02323 opgaver (3 Kap. 1 spm i stedet for two-way)		MSK	NALO
02402 opgaver	30		
02403 opgaver		JKMO	
Tovholder		NALO	
Tentativ tidsplan (justeres af tovholder)			
Deadline 0: Alle opgave udeles på personer	Tentativ tidsplan (justeres af tovholder)		
Deadline 1: 5 spm (på EN) sendes til tovholder	-7 dage	03-05-23	
Deadline 2: Tovholder har færdig EN-udkast klar	-3 dage	10-05-23	
Deadline 2b: Side-spm sendes til tovholder	-1 dage	17-05-23	
Deadline 2c: Alle sæt klar til korrektur	-3 dage	19-05-23	
Deadline 3: Reaktionen fra korrekturlæsere - se fordelir	-2 dage	24-05-23	
Deadline 4: Færdig EN-udkast klar til censor	-7 dage	26-05-23	
Deadline 5: EN sæt klar til oversættelse og facit	-3 dage	02-06-23	
Deadline 5: Hver person oversætter egne opgaver til D	-1 dag	07-06-23	
Deadline 6: Review på den oversatte version (må forde	-1 dag	08-06-23	
Deadline 7: Færdig DA-udkast klar til censor	-3 dage	09-06-23	
Opgaver klar og godkendt	-9 dage	14-06-23	
Eksamensdato		22-06-23	

Figure x. Exam preparation and coordination plan

During a periodic IntroStat meeting, the team members take the responsibility of making questions, corrector/reviewer, and coordinator. For the 02402 course includes Two-way ANOVA and the 02323 course does not. So, for replacing three questions from two-way ANOVA, typically the examination coordinator makes alternative three questions from chapter one for the 02323 students.

Exam question editing and coordination process

Course instructors use relevant R scripts to create exam questions (MCQ), then test and run on R studio and Latex. During this project, the repository has been moved from SVN to gitlab at DTU Compute.

After completing the exam, the course instructors export responses from Digital Exam & List of Participants with R scripts and calculate grade using R studio. All R scripts and R markdown files are stored and shared using SVN and currently gitlab. The generated text file output is copied in the Digital Exam System so that all the 100+ students' grades in each of the two courses can be published at the same time.

4. Discussion

4.1 Piazza for Discussion: Pilot Study in Spring 2023

The discussion feature of the Brightspace LMS (DTU Learn) is not considered user-friendly by the students and was hardly used for asking questions and resulted in receiving too many emails for queries regarding various information, exercise solutions, etc. So, Piazza was used as an experiment during Spring 2023.

Discussions

Discussions List Subscriptions Group and Section Restrictions Statistics

New ▾

More Actions ▾

Filter by: Unread Unapproved

You don't have any discussion topics available to post to. First create a forum by clicking New > Forum and then add a discussion topic to it by clicking New > Topic.

The screenshot below shows Piazza used for 02402 statistics course. All the enrolled students, TAs and the teaching team of six persons were invited to join directly via the emails, informed notified through announcement and introduced during the first day of the course. The students could ask anonymous questions and mainly two instructors responded to the students' queries.

The screenshot shows the Piazza Q&A interface for the 02402 statistics course. The top navigation bar includes links for LIVE Q&A, Drafts, week1(w1), week2, week3, week4, week5, week6, week7, week8, week9, week10, project, exam, logistics, other, week11, week12, and week13. The main content area is titled 'Class at a Glance' and provides a summary of the course's activity. It shows 12 unread posts, 3 unanswered questions, and 2 unanswered followups. A 'Student Enrollment' bar indicates that 338 students are enrolled out of 300. The interface also includes a sidebar with a 'New Post' button and a search bar. The bottom of the interface has links to download the Piazza app from the App Store and Google Play.

Instructors' expectations: The instructors experienced that the students do not try to find all the necessary information on the website, solutions to exercises and previous years' exams, and do not regularly attend class and the exercise sessions. The course instructors encourage the students to attend the exercise session so that they and the TAs can address solutions without investing more time for the large course. Yet, Piazza was adopted to investigate the potential. One of the challenges is that the TAs do not have time between the face-to-face teaching sessions for giving responses to the Piazza questions. However, the intention was also to encourage other students to respond with answers but students' responses to other students' queries were very low.

Regarding the exams and other administrative queries, the students tend to email the course instructors instead of the exam office or study administration. Piazza was expected to reduce direct emails and get peer-group responses, but behavioural change was not observed significantly as the students still sent some emails directly to the teachers. The students started using Piazza during the last few weeks before the exam.

4.2 EdStem for Discussion: Pilot Study in Fall 2023 (also part of Direc P10 project)

In Fall 2023, the EdStem platform was being tested as an alternative to Piazza as R, Python and other codes can be run directly on EdStem, as a part of the editor on the platform.

Instructors' expectations: In large classes, teachers don't have enough time to answer students' questions one by one, especially in terms of code. Teachers have no way to view students' code line

by line, run and test it on their own computer. To provide students and teachers with a more intuitive discussion channel, EdStem is used as an alternative to piazza.

EdStem was expected to reduce direct emails and help students get teachers'/TAs' responses in an organized way.

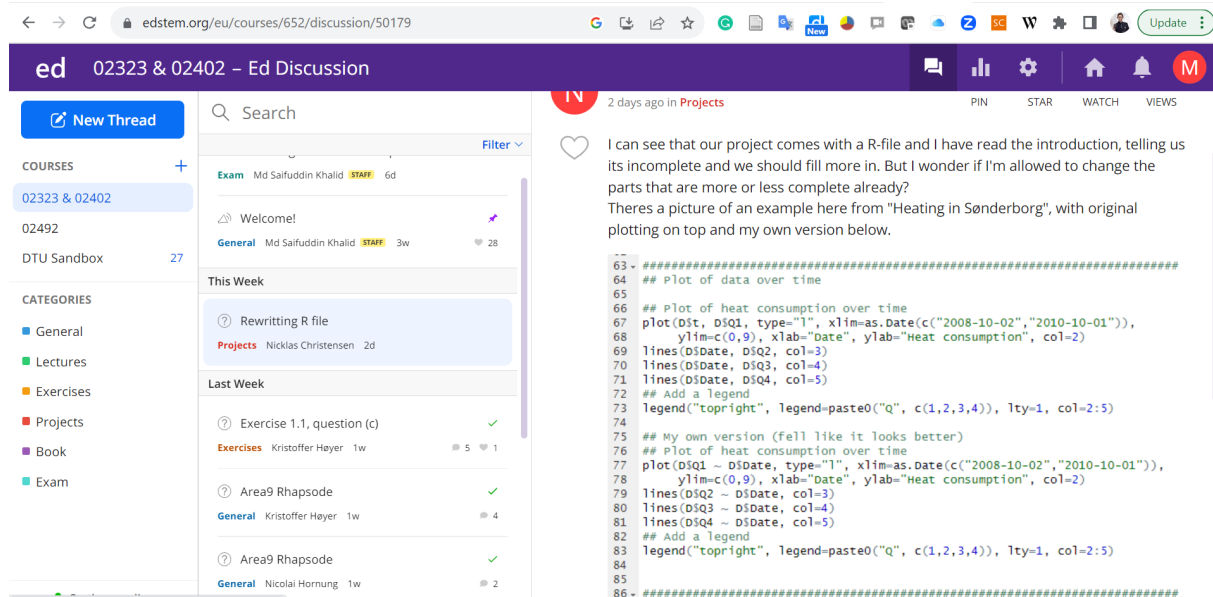






















Figure. EdStem.org (Fall 2023)

4.3 Brightspace LMS (DTU Learn) for Discussion

The screenshots below show LMS (DTU learn) used for project submitting and receiving announcements in the 02402 statistics course. All the enrolled students, TAs and the teaching team of six persons were invited to join directly after they were registered in this course. As the main platform of the teaching activity, Brightspace LMS (DTU learn) can be used from the first day of the course.


Although the LMS has question-answering and discussion functions, students still tend to send emails to teachers to ask questions. The use of LMS mainly focuses on reading course materials, browsing videos, receiving course-related notifications and submitting/correcting projects.


<input type="checkbox"/>	Assignment	New Submissions	Completed	Evaluated	Feedback Published
	Projekt 1  				
<input type="checkbox"/>	Projekt 1: Handel med ETF  Due on 14 March, 2023 11:59 PM 31 January - 16 March		43/320	43/320	43/320
<input type="checkbox"/>	Projekt 1: Varmeforbrug i Sønderborg  Due on 14 March, 2023 11:59 PM 31 January - 16 March		19/320	19/320	19/320
<input type="checkbox"/>	Projekt 1: Skive Fjord  Due on 14 March, 2023 11:59 PM 31 January - 16 March	2	28/320	26/320	26/320
<input type="checkbox"/>	Projekt 1: BMI undersøgelse  Due on 14 March, 2023 11:59 PM 31 January - 16 March		125/320	125/320	125/320
<input type="checkbox"/>	Projekt 1: Genaflevering / Re-handin submission  Due on 11 April, 2023 11:59 PM 15 March - 14 May	23	27/320	4/320	0/320

learn.inside.dtu.dk/d2l/home/145395














02402 Introduction to Statistics Spring 23

This course is open for students
Course setup

Announcements 

Grades have been submitted, info about re-exam 


Peder Bacher posted on 16 May, 2023 4:24 PM

Dear all

We have submitted the grades, so you should soon get them.



In case you have to take the re-exam in June, you must sign up for it here:
<https://forms.office.com/e/PYrp2VT3vL>

We wish you all a good summer and hope you can use statistics to make a good future for us all!


Solution is on the webpage 


Peder Bacher posted on 14 May, 2023 5:34 PM


Dear all

MS Teams Course Connector 


Your instructor may create a course team in Microsoft Teams for communication and collaboration.

Calendar 

Wednesday, July 5, 2023 

Upcoming events 

There are no events to

	Projekt 2				
<input type="checkbox"/>	Projekt 2: Handel med ETF ▼ Due on 11 April, 2023 11:59 PM <u>15 March - 14 May</u>	1	38/320	37/320	37/320
<input type="checkbox"/>	Projekt 2: Varmeforbrug i Sønderborg ▼ Due on 11 April, 2023 11:59 PM <u>15 March - 14 May</u>		17/320	17/320	17/320
<input type="checkbox"/>	Projekt 2: Skive Fjord ▼ Due on 11 April, 2023 11:59 PM <u>15 March - 14 May</u>		22/320	22/320	22/320
<input type="checkbox"/>	Projekt 2: BMI undersøgelse ▼ Due on 11 April, 2023 11:59 PM <u>15 March - 14 May</u>	5	123/320	118/320	118/320
<input type="checkbox"/>	Projekt 2: Genaflevering / Re-handin - Submissions ▼ Due on 02 May, 2023 11:59 PM <u>25 April - 14 May</u>	9	18/320	10/320	0/320

5. Conclusion

The experiment conducted in the courses "02323 Introduction to Statistics" and "02402 Statistics (Polytechnical Foundation)" was initiated with the aim of addressing challenges related to onboarding new teaching roles and Teaching Assistants (TAs). The primary objectives included the creation of a standardized and unified guideline for teachers and TAs and the identification of challenges within the teaching activities of course instructors and coordinators.

Upon examination of the setup, it became evident that onboarding new teacher roles and TAs presented significant difficulties, primarily due to the absence of a standardized guideline. New teaching roles were confronted with the daunting task of navigating through multiple software systems, leading to inefficiencies and potential disparities in their performance.

As a result, it is recommended that the implementation of a unified operational guideline be prioritized. Such a guideline would not only streamline the onboarding process but also serve as a valuable resource for addressing the problems and needs of teachers, as revealed through the experiment. Furthermore, DTU's plan to transition from R to Python in 2024 may introduce variations in software usage, but the workflow derived from this study remain relevant. It is advisable to use the insights gained from the results of interviews to iteratively improve teaching processes and software usage.

In addition to these findings, it is worth noting that this study explored the use of three new software to enhance the classroom interaction experience. These software platforms, including Kahoot (designed to increase classroom engagement), Piazza (a versatile discussion platform), and Edstem (Discussion forum supporting multiple programming language compilation in the web editor), were implemented to try to improve student engagement within the courses.

In summary, it is unlikely to have one large software system and the teachers and TA coordinators will continue to use multiple software systems and underlying infrastructure at DTU. This project attempted to make journey maps, overview of the software systems as part courses' activities, and the guides made for various software use. The aim is to improve the onboarding processes for new teaching roles and TAs in addressing the challenges encountered. The journey map might be further improved by including dates and deadlines, which can be developed as a calendar tool that will remind the course coordinators and instructors for completing the tasks and also remind the software to be used for those.

Acknowledgement

The project "Software Infrastructures for Teaching at Scale" is co-funded by Digital Research Centre Denmark (DIREC) and the technical university of Denmark's Department of Computer Science and Applied Mathematics (DTU Compute).

Appendix I

Interview Guide for interviewing course instructor and coordinator

Interview Questions:

1. What is the name of your course?
2. How many students in one course?
3. Could you detail the process of your tasks?
4. Did you encounter any challenges or limitations in terms of instructional resource allocation or teaching objectives during your tenure as a course instructor/teaching assistant?

5. Could you (instructor/coordinator) give me some instances of how you overcame the challenges?
6. Which software or IT infrastructure instructors do you utilize to address these challenges?
7. What are the outcomes or results of overcoming these difficulties?

Appendix II

Instructions for direction of Introstat projects

This instruction lists the things you just need to know before correcting Introstat projects. If it is the first time you must correct, we will arrange a time to review what needs to be done.

General

- I. Corrected before 12 in the morning on the date at \texttt{Mail stating if you passed Project ...} for the course and project in question:
 - 02323: \url{https://02323.compute.dtu.dk/projects}
 - 02402: \url{https://02323.compute.dtu.dk/projects}
- II. You fill in points for each report on the entry sheet
- III. We send an email on the date at \texttt{Mail stating if you passed Project ...}, which contains whether it was approved, as well as who corrected it, and that they can get further feedback for the group bill the following day
- IV. The students will therefore never know what points they have received, only whether it has been approved or not
- V. You give feedback to the students during their submission in Learn (short text, or annotated in the report (uploaded or directly in Learn)):
 - Go under: "Assignments -> relevant project -> relevant student's submission"
 - Do not write anything about the points given or whether it is approved
 - Write briefly (10-20 lines), as further feedback is given verbally the next day (your name is in the email they received)
 - You can also provide the comments as annotations in the pdf
 - \emph{Press "Save draft" and do not publish}
- VI. We release feedback to everyone at the same time as we send the email if they have been approved.
- VII. The hours given for directions are: 60 min. to start per new project, and 30 min. per project.

Correction instructions

- I. For each project there is a correcting guide
- II. This guide contains an example of what a good report can look like and it states how you should count points
- III. In red, it is described how you can "correct"/evaluate the answers. Don't get bogged down in details, ie. if you judge that the student gives a reasonable answer, even if it does not completely correspond to the sample report, then you give an appropriate number of points!

Entry sheet

Everyone gets an entry sheet in which the points given for each question are recorded. The maximum number of points that can be awarded is entered.

- I. You enter points for each question, we only use these to calculate the total (and not the value in \file{Sum})
- II. You can either write points for each sub-point or total points for each question in the sheet
- III. We do not use \file{Sum} and \file{Passed}, you can change them as you wish.

Set up navigation bar on LEARN

Copy from last semester

- I. See in \file{projects.pdf} on copying the assignments
- II. Not sure if possible to copy the navigation bar layout

Navigation bar layout

- I. In learn go to "course admin" and then "Navigation and Themes"
- II. "Create Navbar"
- III. Add My Course
- IV. Add Announcements
- V. Add Assignments
- VI. Add Custom link to the website

TA Hiring

- I. To the "study administrative coordinator" send hired TA list, which course and if any are half time.
- II. Tell TAs only to report only group exercise hours (total 45.5) in Fusion, we report the assignment correction hours to Pia directly in the end.
- III. Hire TAs ?? This is not the way it works now: Sync the number of hours with Signe Møller Jørgensen smjo@dtu.dk, remember to add hours for exam help days and online help., see "misc/pbac/semester/tim"??
- IV. Start a mail thread with TAs and invite for intro meeting if necessary. Give them also information about registration of themselves as TAs via Anna Jensen annje@dtu.dk.
- V. TA timer online og få booket lokale til eksamensforberedelse

Preparation for the course

- I. Add the DTU week numbers in your calendar and lecture events
- II. Update the website and update project dates, see script for generating the Agendas \file{trunk/website/make.R} (and \file{project.pdf})
- III. Update the participant list on the websites (Admin/Course) by copying in the Inside url for the current course
- IV. Add project todo (mail TAs, distribute, mail, etc.) reminders in your calendar
- V. Update CN frontpage text (very simple, just refer to the website)
- VI. Copy slides to a new folder. Update the semester date in 'slides/slidesTitle1.Rnw' (It appears in the lower right footer of the slides), and 'slides/slidesTitle2.Rnw'
- VII. Add the TAs to the CN courses
- VIII. Set remove answers to all weeks and makeAllSlides.R in the folder

A simple check list with todo for each week

Week 0

- I. Send welcome message (link), include:
- II. Website, Agendas, read first chapter
- III. Send TA information about first time and how they should distribute in the rooms during group exercises
- IV. Make all project assignments in CN, copy from last year, see \file{projects.pdf}

Week 1

- I. First day in the week: Send welcome message (link, maybe two messages one in beginning of the week and one the day before), include:
- II. Website, Course Material, Agendas, Testquizzes, ...(see last years message)

- III. Find the students with projects already approved with the script trunc/eksamen/projectFindFreebees.R and mail them and a message to all (find texts in \file{projectTexts.txt})
- IV. Update CN with projects.
- V. Project message: introduce it in the lecture and send it there. Simple guidelines with link to the website project page. Emphasize that plagiarism will be taken serious.

Week 6

- I. Launch mid-term evaluation
- II. Write to the TAs for their preferences regarding type and amounts of project 1 evaluations

Week 7

- III. Check mid-term evaluation and include in lecture
- IV. Project 1 hand-in
- V. Check for plagiarism
- VI. Distribute the reports to TAs after plagiarism check

Week 9

- I. Write to the TAs for their preferences regarding type and amounts of project 2 evaluations

Week 10

- I. Project 2 hand-in
- II. Check for plagiarism
- III. Distribute the reports to TAs after plagiarism check
- IV. Trial exam in CN:
- V. Create task
- VI. You just have to create right??
- VII. Import Previous Exam....

Week 11

- I. Set up evaluation

Week 13

- I. Examination announcement
- II. Set up a digital test exam
- III. Submit a report in Campusnet (Copy some of the good and bad of the evaluation)

Week 14

- I. Exam announcement
- II. Remember to write your answers on paper
- III. Remember that you can give more than one answer, but then it counts as 'don't know'
- IV. Remember to check where you will sit on the portal

Exam

EXAMINATION (Pass in all courses that have run (02402, 02323, 02593, 02403)):

- I. Create a new exam task and select 'Multiple choiche task'
- II. Set the start time
- III. Select 'Use custom score calculation'
- IV. Copy the description from the previous exam
- V. Edit first question, upload real files and write their names
- VI. Insert the correct answers

- VII. Create corresponding exam Copies for the other courses
- VIII. Make sure that those from 02593 do not get grades reported in 02323.

On streaming

Stream in Aud. 42

Setup in 42:

- I. Set the slides on the projector(s)
- II. Make the sound well, if feedback, then lower the level on the mic
- III. To share room choose "Share room" and set "Public"
- IV. Remember, for the monitor to be shared in the other room **non-mute the loudspeaker icon on the monitor symbol**
- V. Then go to the other room
- VI. In the other room and pull "shared room" into the monitors and select "Room 42"

Feedback to students

- I. If everything is really good, then short feedback that it is really good.
- II. The points in the help form can e.g. be used to narrow down the points you can give feedback on.
- III. If a lot of work has been done on it, but there is a lot wrong, then we give more feedback (they can learn something), but still briefly in writing, but they can then get more elaborate orally for group calculation the next day.
- IV. For oral feedback, you can open their submission and go through the points you have commented on, then they can ask questions and you can explain what they have not understood.

Projects

Create the projects in Learn

- I. Go under "Course Admin -> Import / Export..." and copy the Assignments from last semester
- II. Change the dates afterwards under Assignments.
- III. Remember to get the TAs preferences (share sheets where they can type in their preferences for both projects, like which one and how many they want to correct)
- IV. Remember to check under "Submission & Completion": "Only the most recent submission is kept."
- V. Set the allowed file formats to ".pdf,.r"

Updating projects .z files

- I. Run `\file{svn update}` in the `\file{../projects}` folder, such that all projects files are updated
- II. Run `\file{../projects/make_all.R}` and cross your fingers that it runs ;D
- III. Run `\file{../projects/zipFiles/makeZipFiles.R}` to update all `\file{.zip}` files

Sharepoint folder

At the start of the semester, create a sharepoint folder (same folder for 02323 and 02402). See separate guide.

Project direction wishes

- I. Ca. three weeks before deadline of project 1:
- II. Create an excel sheet with preferences using template
- III. Upload to sharepoint folder, ask TAs to fill in their preferences.

- IV. Pre-meeting about project correction for new TAs, suggested day: Monday before deadline of project 1.

Distribute projects to TAs

- I. TODO: Copy the project folder from the template (see 'taProjectsAdmin/Folder_Template') including all the files.
- II. TODO: Update 'TAProjects.xlsx' to include all the TAs from the two courses
- III. Close the assignments on Learn (use bulk edit and set 'end date')
- IV. Download all reports in .zip files.
 - For each assignment, remember to check that all are shown in the list (check below the list, set to '200 per page'. If there are more, then you have to download in multiple .zip files)
 - Check that no file submissions are missing a name (e.g. '.RData' files will cause an error)
 - Unzip the zips into the assignment folders (e.g. there will be an "Input/Reports/skivefjord/287623- 23423 - s123456 name/report.pdf" file)
 - Delete the zip files
- V. Open "MakeSheets.R"
- VI. Run the first part where data is read. The following might occur:
 - If TAProjects.xlsx is not found, then look how it should be from last semester
 - If while reading the reports a DTU initial is found, then follow the instructions and add to the 'initials_and_studienr.csv' file
- VII. Now the reports must be distributed among the TAs according to their preferences, so do that in the script
- VIII. Write the sheets: THEY ARE NOW THE REFERENCE (or actually the ones that will be downloaded when filled, but these here will be checked against the downloaded to see if all was corrected), so any change in the distribution must be made directly in the sheets. Note that they will not overwritten if they exists already.
 - Un-comment the line where the file is produced (line 91)
 - Remove the '_REMOVETHIS' text from the path string
 - Run the line – Put the '_REMOVETHIS' text back in to the string
 - Comment the line
- IX. Write the zip-files.
- X. Upload the sheets and the zip-files to share them with the TAs.
- XI. Send the TAs a message, see last semester for the content.
- XII. Check into svn: REMEMBER not to include the downloaded reports!

Project results

The TAs have a deadline 12.00 for correcting reports the day before the lectures. Project results shall be done on the same day:

- I. Download rettemark from sharepoint into a folder (e.g. "indtastningsark_retur")
- II. Run processResults.R. See details in the file.
- III. Send out emails to students about passed/not passed via script in processResults.R. DO CHECK EVERYTHING IS CORRECT AND MAKE A TEST EMAIL before sending out emails to students. Please also save the file with *test = TRUE* on. **Note: emails can only be sent when you are connected to DTU's wired network!**
- IV. Publish all feedback on Learn

Update the project status list

(\file{projects.csv})

Run \file{eksamen/projectsUpdate.R}

Find students who don't need to hand in a project

- I. Download the participants list for each course from: `\url{http://deltagerlister.ait.dtu.dk/}`
- II. Find the course and take "datafil", mark all (with the header), and then paste into a text file
- III. "Save as" under current `\file{eksamen/"date"}`
`\file{downloadedLists/deltagerlisteXXXXX.csv}`. IMPORTANT, simply use notepad or another texteditor, not Excell, to keep the format.
- IV. Run the `\file{projectFindFreebees.R}` to make an email list
- V. Send a mail and message to in the beggining of the course, see `\file{calendar.pdf}` under week 1
- VI. Find texts in `\file{projectTexts.txt}`

Find who didn't pass the projects and must be signed off the exam

- I. After finalizing all points from the TAs evaluation of the reports run the `\file{projektResultater.R}` in the semester folder
- II. Then run `\file{eksamen/projectsUpdate.R}`
- III. Update the deltagerliste (like in done in the previous section). USE "Eksamenstilmeldinger" (exam registrations)
- IV. Run the `\file{projectMakeAfmeld.R}`, it makes `\file{afmeld.txt}` in the semester folder, which is a list of who to sign off the exam
- V. Send a mail to the ones to be de-assigned: a mail text is in `\file{projectTexts.txt}`
- VI. Send the list to *Henriette and Joan*

Add students who previous had the exam before 2014 fall

- I. Open `\file{ ../eksamen/misc/freebeeslister/freebees.csv}` and in the bottom add the student
- II. Run `\file{projectsUpdate.R}` to update the `\file{projects.csv}` (NOTE don't edit in `\file{projects.csv}`)

Developing project

File naming conventions

- I. Each project has an id consisting of a name and the project number (1 or 2)
- II. Each project has a folder with its id (e.g. skivefjord1)
- III. See the naming of each file in the skivefjord folder

Project description conventions

- I. Each project has a description in both Danish and English
- II. See the format in skivefjord1_dansk.pdf
- III. In each project folder is a "make.R" containing the R script to knit and compile the document

Compiling zips for upload

- I. The R-script "zipFiles/makeZipFiles.R" generates zip files for upload on CN.
- II. Zip files for upload are placed in "zipFiles/"

Weeks for deadlines (works for fall, but usually needs to be modified for 02323 in spring)

Find the weeks from: `\url{http://www.dtu.dk/Uddannelse/Kursusbasen/Undervisningsaaet}`

- I. Find the exam date from (also lokaleoversigt should be there at Portalen):
`\url{http://www.dtu.dk/Uddannelse/Kursusbasen/Eksamensskema}` or
`\url{http://portalen.dtu.dk/DTU_Generelt/AUS/Studerende/Infosite/Eksamensdatoer.aspx}`
- II. Add the weekdates and examdate to your calendar

Project 1:

- I. Week 7 hand in (must be one week after week 6)
- II. Week 9 mail and feedback
- III. Week 10 re-handin

Project 2:

- I. Week 10 handin (must be one week after week 9)
- II. Week 12 mail and feedback
- III. Week 13 re-handin

Previous semester dates

Template to be re-used for scheduling deadlines for project dates

2016dec

Project dates 02402 (Tuesdays) dates for Project 1:

- I. Oct.: Project 1 hand-in (23:59:59)
- II. Oct.: Mail stating if you passed Project 1 and which TA evaluated it
- III. Nov: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- IV. Nov.: Re-hand in deadline of Project 1 (23:59:59).

02402 (Tuesdays) dates for Project 2:

- I. Nov.: Hand-in deadline of Project 2 (23:59:59)
- II. Nov: Mail stating if you passed Project 2 and which TA evaluated it
- III. Nov: Get feedback on Project 2 during the exercises from the TA that evaluated your report
- IV. Nov: Re-hand deadline of Project 2 (23:59:59)

02323 (Fridays) dates for Project 1:

- I. Oct.: Project 1 hand-in (23:59:59)
- II. Nov.: Mail stating if you passed Project 1 and which TA evaluated it
- III. Nov: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- IV. Nov.: Re-hand in deadline of Project 1 (23:59:59).

02323 (Fridays) dates for Project 2:

- I. Nov.: Hand-in deadline of Project 2 (23:59:59)
- II. Nov: Mail stating if you passed Project 2 and which TA evaluated it
- III. Nov: Get feedback on Project 2 during the exercises from the TA that evaluated your report
- IV. Dec: Re-hand deadline of Project 2 (23:59:59)

2016may

02402 (Tuesdays) dates for Project 1:

- I. Mar.: Project 1 hand-in (23:59:59)
- II. Apr.: Mail stating if you passed Project 1 and which TA evaluated it
- III. Apr: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- IV. Apr.: Re-hand in deadline of Project 1 (23:59:59).

02402 (Tuesdays) dates for Project 2:

- V. Apr.: Hand-in deadline of Project 2 (23:59:59)
- VI. Apr: Mail stating if you passed Project 2 and which TA evaluated it
- VII. Apr: Get feedback on Project 2 during the exercises from the TA that evaluated your report

VIII. May: Re-hand deadline of Project 2 (23:59:59) 02323 (Fridays) dates for Project 1:

- IX. Mar.: Project 1 hand-in (23:59:59)
- X. Apr.: Mail stating if you passed Project 1 and which TA evaluated it
- XI. Apr.: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- XII. Apr.: Re-hand in deadline of Project 1 (23:59:59).

02323 (Fridays) dates for Project 2:

- XIII. Apr.: Hand-in deadline of Project 2 (23:59:59)
- XIV. Apr.: Mail stating if you passed Project 2 and which TA evaluated it
- XV. Apr.: Get feedback on Project 2 during the exercises from the TA that evaluated your report
- XVI. May: Re-hand deadline of Project 2 (23:59:59)

2015dec

Project dates 02402 (Tuesdays) dates for Project 1:

- I. Oct.: Project 1 hand-in (23:59:59)
- II. Nov.: Mail stating if you passed Project 1 and which TA evaluated it
- III. Nov: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- IV. Nov.: Re-hand in deadline of Project 1 (23:59:59). 02402 (Tuesdays) dates for Project 2:
- V. Nov.: Hand-in deadline of Project 2 (23:59:59)
- VI. Nov: Mail stating if you passed Project 2 and which TA evaluated it
- VII. Nov: Get feedback on Project 2 during the exercises from the TA that evaluated your report
- VIII. Dec: Re-hand deadline of Project 2 (23:59:59) 02323 (Fridays) dates for Project 1:
- IX. Oct.: Project 1 hand-in (23:59:59)
- X. Nov.: Mail stating if you passed Project 1 and which TA evaluated it
- XI. Nov: Get feedback on Project 1 during the exercises from the TA that evaluated your report
- XII. Nov.: Re-hand in deadline of Project 1 (23:59:59). 02323 (Fridays) dates for Project 2:
- XIII. Nov.: Hand-in deadline of Project 2 (23:59:59)
- XIV. Nov: Mail stating if you passed Project 2 and which TA evaluated it
- XV. Nov: Get feedback on Project 2 during the exercises from the TA that evaluated your report
- XVI.** Dec: Re-hand deadline of Project 2 (23:59:59)

Administration of projects in Introstat

Distribute project to TAs

- I. TODO: Copy the project folder from last semester (see '../misc/pbac/2021maj/project1/')
 - II. TODO: Run 'clean.R' in the folder
 - III. Close the assignments on Learn (use bulk edit and set 'end date')
 - IV. Download all reports in .zip files
 - For each assignment, remember to check that all are shown in the list (check below the list, set to '200 per page'. If there are more, then you have to download in multiple .zip files)
 - Unzip the zips into the assignment folders (e.g. there will be an "Input/Reports/skivefjord/287623-23423 - s123456 name/report.pdf" file)
 - Delete the zip files
 - V. Open "MakeSheets.R"
 - VI. Run the first part where data is read. The following might occur:
 - If TAProjects.xlsx is not found, then look how it should be from last semester
 - If while reading the reports a DTU initial is found, then follow the instructions and add to the 'initials_and_studienr.csv' file
 - VII. Now the reports must be distributed among the TAs according to their preferences, so do that in the script.
 - VIII. Write the sheets: THEY ARE NOW THE REFERENCE (or actually the ones that will be downloaded when filled, but these here will be checked against the downloaded to see if all was corrected), so any change in the distribution must be made directly in the sheets. Note that they will not be overwritten if they exist already.
 - IX. Write the zip-files.
 - X. Upload the sheets and the zip-files to share them with the TAs.
 - XI. Send the TAs a message, see last semester for the content.
 - XII. Check into svn: REMEMBER not to include the downloaded reports!

Exam

Set up of exam

1. Log in: \url{eksamen.dtu.dk} and go to the current semester, where the exams should be listed.
2. Log in as administrator
3. Select the exam to set up.

Set up only for Test Exam

For the test exams we need to insert dates, not for real exams (exam office controls those, also time for special needs etc. is added automatically)

- I. Go to Exam Data in the left pane,
- II. Exam Start: Right when you are editing
- III. Exam End: The night before the actual exam
- IV. Assessment start and end does not matter.
- V. Don't change other field and save.

Under "Assessors" the ones who must be able to access the answers (logging in as assessor) must be added.

- I. In test exams we have to first add us as assessor. Click directly on the "Edit assessors" field and find your name (as Examiner)
- II. For test and real exam: To mark all students click the assessor's name in the column and click save!

Add multiple choice

(FOR TEST EXAM, remember to use the previous semester files!)

- I. Go to the "Multiple Choice" tab (or "Exam question set" and click "Change to multiple choice") and take "Edit multiple choice"
- II. In the bottom "Create questionnaire". Select "Manual scoring". DO NOT select "Jeg ønsker at spørgsmål vises i tilfældig rækkefølge for studerende" and "Jeg ønsker at svarmuligheder vises i tilfældig rækkefølge for studerende".
- III. Delete the first question (under dots on the question)
- IV. Give the questionnaire the correct name (e.g. \file{02323 Prøveeksamen / Test exam})
- V. Upper right dots: "Paste questions": Copy the content of the generated text file named (e.g. \file{questions_2020dec_02323_solution_en.txt} generated in the exam folder)
- VI. Replace the default text with content from the file \filelink{guides/misc/examText.txt}
- VII. Replace the pdf file (links): Click the "link" button in the menu above.
- VIII. Go to "Upload", choose the exam pdf file and "Send it to the Server"
- IX. Give it the correct name and "Ok".
- X. Remove the old .pdf names.
- XI. FINALLY, under the "Multiple Choice" tab, select "Add multiple choice" and select the questionnaire and press save.
- XII. For test exam, remember to make it visible and copy the pin code or deactivate it on "Exam data" tab.

How to change multiple choice:

The question sheet cannot be changed after being attached to an exam. However the following works:

- I. Go to the "Multiple Choice" tab, remove the attached multiple choice
- II. Go to "<https://designer.mcq.eksamen.dtu.dk/>". The question sheet can now be changed.
- III. Go back to the "Multiple Choice" tab and re-add the question sheet.

Set up for the additional exam

The additional exam for notes must be set up the same way regarding dates and assessors. In addition:

- I. Select the exam
- II. Set dates (only for test exam)
- III. Go to the "Set of exam questions" tab (or "Multiple Choice" and click "Change assignment type") and take "Add set of exam questions"

Set:

- I. Title: \file{Mellemregninger / Notes}
- II. Description: \file{Aflever dine mellemregninger her / Upload your notes here}
- III. Assign the assessors

On the exam day

- I. The central administration configures the start and end time in the digital exam system. The exam office also provides hand-written exam answers for some students and in such a case the hand-written document is considered for assessment and the digital exam system's entries are excluded.
- II. After the exam, the examiner uploads the questions and the solutions to the courses websites and sends email to the students through the learning management system's announcement feature.

Calculating the grades after the exam

Prepare the folder:

- I. Copy the \file{resultater} folder from a previous year
- II. Copy the correct answer files
(e.g. \file{correctAnswers_2017aug_02323_02402_solution_en.rda}) to the \file{resultater} folder.

Download the digital answers

- I. Go to \file{exam.dtu.dk} and find the exam (remember to add you as assessor to each student, see above).
- II. Select the exam and take "Go to MCG overview" and select "Export".
- III. Save the file in \file{resultater/examLists/}.
- IV. Convert to ".csv" with ";" as separator.

Download three lists for the course

- I. Download the \file{Eksamenstilmedinger} list for each course from:
\url{http://deltagerlister.ait.dtu.dk/}
- II. Click \file{Datafil} in the upper menu
- III. Select all (with the header) and copy-paste into a text file
\file{examLists/deltagerlisteXXXXX.csv} (REMEMBER to open and save this one with ";" separator!)
- IV. Download the \file{Eksamenstilmedinger} list for the course from:
\url{http://deltagerlister.ait.dtu.dk/}
- V. Click \file{Datafil} in the upper menu
- VI. Select all (with the header) and copy-paste into a text file
\file{examLists/deltagerlisteXXXXX.csv} (REMEMBER to open and save this one with ";" separator!)
- VII. Go to \url{inside.dtu.dk} and "Education" and under "toolbox" take "Marks"
- VIII. Select the right course, and select "Import/Export data" and then "Export to Excel"
- IX. Save the file in \file{resultater/examLists/karakterlisteXXXXX.xml}, where \file{XXXXX} is course number

- X. Convert the saved files to `\file{karakterlisteXXXXX.csv}`, with formatting (`\file{;}` and quotes on strings)
- XI. Remove the `\file{.xml}` files

Update the files

- I. Run the `\file{trunc/eksamen/projectsUpdate.R}` to update the `\file{project.csv}`
- II. Go and find the green envelopes, which contain answers delivered on paper and no digital hand-in code (digital hand-in counts over white paper hand-in)
 - Only answers on white paper, which are clearly stated should be typed in
 - (REMEMBER to open and save with ";" separator!) In `\file{resultater/examLists/deltagerlisteXXXXX.csv}`: Type in the answers in additional three columns in (answer 1-10, answer 11-20, answer 21-30), as integers `\file{xxxxxxxx}`, with `\file{6}` as no answer
- III. Make a check that the students were actually there
 - Find the EM lists and in `\file{resultater/examLists/deltagerlisteXXXXX.csv}` type in 'e' in column "K", then it's checked in the scripts, that they didn't hand in.

Calculate the grades with \file{analyse.R}

- I. Read the data: include each course:
 - Digital answers count over paper answers.
 - If a student have both answers on paper than on digital and they are different, these are printed out.
 - You can use the paper answers by: `\file{res[res\${studienr}=='s123456', qnames] <- respaper[respaper\${studienr}=='s123456', qnames]}`.
- II. Set the path to load the correct answers.
- III. If any questions are to be taken out of the evaluation, specify the question numbers in `spmOut` (otherwise set to NA).
- IV. Set the `\file{limit}` to pass and limits for all grading and calculate.
- V. Check for the projects, if anyone has sneaked below the rader and have a grade, without having their projects approved.
- VI. Write the results and go to `\file{Marks}` under CN.
 - Note that in the script students missing on the karakterliste are printed.
 - Upload the results by copying directly from `\file{resultsCN_XXXXX.txt}`.

Appendix III

Links to Video guide

Video guide on how to distribute projects to TAs.

[02323 Project 1 division-20231016 103438-Meeting Recording.mp4 \(for DTU's internal use only\)](#)