XM: A Customised Software Tool to Facilitate Learning and Professional Skills Development in Operating Systems

Alina-Delia Călin[a], Rareș Florin Boian[b] and Horea-Bogdan Mureșan[c]
Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu Street, 400084, Cluj-Napoca, Romania

Keywords: Operating Systems, UNIX, Teaching Tools, Evaluation Tools, Class Management, Cheating Detection.

Abstract: In this paper, we present the XM tool and a novel perspective on teaching Operating Systems as an undergraduate course, focused towards the development of professional skills. We propose a modular content course plan and a suite of software tools that are specifically designed to support academic activities that include learning, assessment, cheating detection, and attendance management. With this approach, we are able to provide students with access to a better educational environment that will support their learning experience and prepare them for real-life industry scenarios. For teachers, it provides more efficient class management for grading and attendance tracking, allowing more time to spend on personalised feedback for students.

1 INTRODUCTION

From textbook content to online resources used in teaching Operating Systems (OS), most course materials are highly theoretical and abstract (Kankuzi, 2019). While it is a fundamental component of all undergraduate computer science degree programmes, the allocated time and resources allow for a deep and detailed study of only the most important concepts. Expanding on this requires approaches and tools that support the demonstration of OS concepts in the context of a real-world, live scenario, such as using command-line tools for file system navigation, compilers, but also managing processes, resources and users (Vavrecková, 2020).

In this paper, we propose a dynamic approach to teaching OS by utilising a suite of software tools designed to offer a great hands-on learning experience. This is performed with the help of live programming and interaction with the Unix operating system, all the while improving the detection and prevention of cheating, even in online environments. To evaluate the impact of the customised software system on learning and teaching, we formulate and answer three specific research questions, presented in Section 3.

[a] https://orcid.org/0000-0001-7363-4934
[b] https://orcid.org/0000-0002-0993-8842
[c] https://orcid.org/0000-0003-4777-7821

2 BACKGROUND

In terms of teaching operating systems, there are many attempts to shift the highly theoretically oriented courses towards a more practical approach, using tools and focusing on understanding the OS concepts in a hands-on experience. A few decades back, the various operating systems used were much simpler, and the OS course was oriented towards the senior-level students or graduates. However, in time things became more complex and with multiple types and generations of systems, the theoretical concepts became more abstract and disjoint from the experiential part, with heavy content (Weina et al., 2018).

In this context, studies (Weina et al., 2018) emphasise that students often have difficulties linking this course to others, as well as replicating in experiments some key concepts (such as memory management, concurrent behaviour). To solve these issues, they use a related thinking principle of teaching, providing a simple OS to demonstrate aspects, invite students to learn through experiments, and make associations. Another paper (O'Brien, 2017) describes an approach using the Linux SystemTap monitoring tool to capture and illustrate specific operating system concepts such as scheduling, file system implementation, and memory management. This tool offers an additional simple and valuable resource to solidify the understanding of theoretical concepts.

Virtual machines have also been widely used to provide a safe learning environment to students in OS

Călin, A., Boian, R. and Mureşan, H.
XM: A Customised Software Tool to Facilitate Learning and Professional Skills Development in Operating Systems.
DOI: 10.5220/0012590300003693

In *Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024) - Volume 1*, pages 330-337
ISBN: 978-989-758-697-2; ISSN: 2184-5026

courses. Using Linux as the production operating system and VMware as the platform, a virtual kernel development lab was created, covering useful subjects including POSIX Threads, ext3 file system, and kernel patch application (Nieh and Vaill, 2006). The EduFUSE tool was developed, extending an existing framework to support both the C and Java languages (Thompson et al., 2020), compatible with Linux and MacOS, to study and learn about file systems.

Recent approaches in computer science are based on browser client-side web technologies, with almost any programming language supported, even the basic Linux commands (Sharrock et al., 2022). While they provide simple access to a practice terminal, they skip on the very tedious process of setting up the required environment. Also, the problem of cheating is not properly addressed in this case. Bailey et al. present a tool named uAssign, aimed at teaching and self-assessing Unix command line skills, based on a web-embedded terminal (Bailey and Zilles, 2019).

Another aspect of teaching is related to class management, student assessment, and grading, in which cheating prevention is a major topic of interest (Vykopal et al., 2022). This led to the development of an Automatic Problem Generation software (Fowler and Zilles, 2021) for creating personalised tasks for each student (to prevent cheating) for a hands-on computer lab environment, for a cybersecurity course. It involves creating arbitrary networks and virtual machines and then extending the tool to an open-source platform of exercises and competitions.

3 PROPOSED COURSE APPROACH. CASE STUDY

The Operating Systems course is part of the first year, second semester bachelor's degree curricula for the Computer Science study undergraduate programme from a higher education institution. It is a mandatory course in the 3-year computer science programme offered by the university. The total number of students enrolled each year is around 400, of which half study in their native language programme and half in the English teaching programme. There are no mandatory prerequisites, but basic knowledge of C programming is recommended (and gaps are being addressed in the course planning and laboratory practice).

The objectives of the discipline are focused on learning the basics of Unix commands, shell programming, and the standard C system API. Students will learn to interact with an operating system beyond the standard user interface, using the command line. At the end, the students will have a good understanding of processes (scheduling, concurrency, synchronisation, deadlocks), memory management (allocation, loading, placement, replacement, caching), file system organisation (including data corruption prevention, existing file systems and their applicability), Linux command line (frequently used commands, Sh/Bash programming), C file API (process creation and inter-process communication API), POSIX threads, and synchronisation mechanisms.

In this context, we introduce a set of software tools, of which the XM tool is specifically built to address the requirements and objectives of the Operating Systems course, from the perspective of both students and teachers. We assess the impact of our approach, by answering the following research questions (RQ):

- RQ1: Does XM simplify grading and attendance management for the teacher?
- RQ2: Does our approach facilitate the learning of practical skills for industry by students?
- RQ3: What is the impact of our cheating detection mechanism?

3.1 Modular Approach of the Course Contents

The course structure and syllabus were developed in line with the initial aims of putting practical skills and competencies first and theory second, and willing to keep students engaged throughout the entire semester.

In terms of selecting and planning the course content for seminaries and laboratories, we used sprints of 2-3 weeks for each chapter, followed by a formative evaluation, as described in Table 1. The lectures follow a similar planning, providing the initial introduction to these concepts (which are then detailed and put into practice at the seminar and lab) and then moving on to detailed theory about operating systems starting from weeks 10-11 onwards.

In addition, many chapters are based on previously acquired knowledge and competencies, thus this approach ensures greater participation of students. Each chapter will end with a practical laboratory evaluation in which students must demonstrate their acquired skills and be individually assessed and supported early on if they do not have the minimum required skills to participate in the next chapter.

Continuous evaluation in the form of short tests that measure basic learnt skills is more useful to both students and teachers in adapting the rest of the content based on what was learnt and what has not yet been handled well enough. In this sense, planning can undergo slight changes, and the teacher can choose to include more examples of gradual difficulty.

Table 1: Course contents by modules.

Chapter sprint	Contents for seminar and laboratory
Weeks 1-3:	- UNIX command line introduction. C programming in the UNIX command line - gcc, system functions, system manual. Detecting memory problems with valgrind.
Evaluation:	Test on server (TC): debug a C program.
Weeks 3-5:	- Regular expressions. Basics of grep, sed, awk, find, sort, uniq, cut.
Evaluation:	Test on server (TGSA): solve a text processing task using regex and several unix commands.
Weeks 6-8:	- UNIX Shell programming - basic syntax, I/O, redirects, commands combination. Permissions, use of /etc/passwd and other system files.
Evaluation:	Test on server (ESh): write a Shell program to solve system specific tasks (involves files, directories, processes, users).
Weeks 9-11:	- UNIX Processes: fork, wait, exec, signal, kill, exec. UNIX Signals. IPC: pipe, FIFO, dup2, popen.
Evaluation:	Test on server (EPr): write a C program to solve a problem that use inter-process communication between several processes.
Weeks 12-14:	- POSIX threads, mutexes, RW Locks, conditional variables, semaphores, barriers. Programming examples with synchronisation and deadlock prevention.
Evaluation:	Test on server (ETh): write a C program to model and solve a problem using threads and efficient synchronisation mechanisms.

3.2 Software Tools

Our platform of choice for teaching the OS course is Linux, which is historically the platform of choice of numerous schools teaching the same class. We also chose Linux due to its pervasiveness in the production systems used by the local industry, and its accessibility to students for home practice. Besides connecting over SSH to a remote Linux server, students with Windows computers can use Windows Subsystem for Linux while students with Apple computers already have in MacOS, a Unix based OS.

One of the course goals is to teach the students to use an operating system beyond the usual visual UI, in other words, we want them to get familiar with the command line and learn to harness its power. To achieve this goal, we chose to organise the entire course around a Linux VM (2 vCPUs, 8GB RAM, 20GB SSD), to which the students would connect via SSH and “spend the entire semester in the command line”. As this makes for a great learning environment where students can have a hands-on experience with a multi-user UNIX environment, it became apparent that both teachers and students would benefit if it would also provide means for the other activities involved in teaching: evaluation, grading, attendance taking, proctoring, automatic grade calculation, student status/grade reporting, etc. This led to the design and development of a software tool named XM meant for examination, but was later extended with all the features listed above in year 2020.

3.2.1 The XM Tool

The guiding principles for the development of the XM tool as a support for teachers and students were:

- Purely command line to promote learning of practical programming skills for students
- Assist teachers with class management to reduce the time spent on non-learning tasks (e.g. attendance taking, problem assignment, grading)
- Increase the difficulty of cheating and provide a mechanism for detecting cheating attempts

The tool was developed on the Linux platform, to fully support functionalities required for semester activities such as attendance, evaluation and grading, in a safe and cheating proof environment (see Figure 1).

The implementation language used is Go, together with an SQLite database and is compatible with any Linux distribution, as it makes use of automated user account generation and management features. The development principles are based on the Linux philosophy ”write programmes that do one thing and do it well” (Raymond, 2003) and rapid development, as a user-friendly command line tool.

3.2.2 Managing Student Assessment Activities

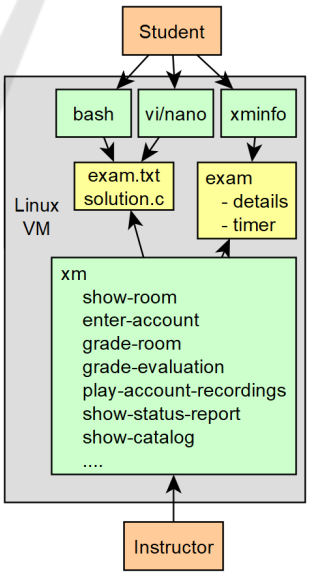

Figure 1: Architecture of the XM Tool.

The XM design for evaluation and cheating prevention is based on the use of one-time operating system user accounts. Students do not have personal accounts on the server; a new account is created for any activity requiring server access. The connection credentials are sent by email to the students and the accounts are locked after the activity is completed. The one-time account acts as a frozen persistent reference of the student's work, allowing for grading and reviewing/regrading of their solutions.

The accounts are configured at the time of creation to record all user activity by starting upon every login using the *asciinema* tool (Kulik, 2022). The console activity recording can be replayed (Hassan, 2020b) later on for ensuring authentic practical assessments (Hassan, 2020a) as it allows easy detection of code pasting or attempts to fetch material from outside the examination account. During the examination, the instructor can monitor the student activity as a summary of processes and connections, or can see details and act using super-user privileges.

In order to assign evaluation sets for an activity that will be graded, the instructor can attach the set(s) when the room for that activity is created (Figure 2) or at a later point (Figure 3).

```
> xm create-room --course os --room roomname
--activity lab
--evaluation-list eval-set-1,eval-set-2
```

Figure 2: Create room.

```
> xm add-evaluations-to-room--room roomname
--evaluation-list eval-set-1,eval-set-2
```

Figure 3: Add evaluation type to the room.

The evaluation sets are stored in a predefined folder structure where only instructors have access. The XM tool, for each account linked with the activity, will randomly choose one problem from each of the specified evaluation sets and will copy them into the home folder of the account, one file per problem. Upon login, the account owner will be able to access the files containing the problem statements and solve them. This way we ensure each student has a unique time-limited task to prevent copying solutions.

3.2.3 Attendance Management

Some of the course activities such as seminars and laboratories require attendance taking as faculty regulations impose a minimum attendance without which the class is automatically failed. The students participating in a course activity are added in an XM room, which acts as an attendance record. Adding the students to a room has been designed to allow the professor to quickly select the students from previous activities, or search for new students by name or ID. In case of online teaching, a simple copy-paste of the attendance list from the conferencing application (e.g. Microsoft Teams, Google Meet, Jitsi, etc) into XM will result in XM tool matching the students by name and add them automatically to the room and asking for user input in cases that it cannot solve, as shown in Figure 4. The instructor does not have to memorise the command line options as XM provides Bash auto-completion, and for every missing argument it presents the user with a single/multi-choice interactive option selection, as seen in the last three lines of Figure 4. With this approach, attendance taking for a seminar of 30 students is much faster.

```
> xm create-room --course os --room roomname
--activity lab
Paste students on separate lines, then press Ctrl-D
john smith
jane doe
adam
^D
Added "john smith" to room "abc" of course "os"
Added "jane doe" to room "abc" of course "os"

Please identify student "adam" in the list below
Select student record for "adam"
  > adam jones
    adam carter
```

Figure 4: Performing attendance.

3.2.4 Facilitate Grading Activity

The grading is performed by entering each one-time student account, evaluating the solution, and then assigning a grade. To simplify the navigation between accounts and grade assignment, XM provides room-wise (often used in practical evaluation) and subject-wise grading features. The subject-wise grading is used for the written summative evaluations which consist of 10 open-answer questions assigned to 150-200 students at a time. Both approaches allow the examiner to navigate among the student solutions in a next/previous manner.

Grading consistency and efficiency are much easier achieved if the evaluator does not need to switch from one problem to another, but rather focuses on all solutions for the same problem. Reducing the examiner's context switching when grading the practical evaluations does not have much impact, since the number of distinct problems is small. For the summative evaluation, context switching minimisation is significantly more important as evaluators prefer to grade consecutively 150 solutions to the same problem before changing context (problem statement), instead of grading each student's 10 solutions to the 10 different problems before moving to another student. In Figure 5 is a sample grading session where the evaluator as-

signs a grade, navigates to the next student, and there reconsiders the grade and changes it for a higher one.

```
> xm grade-evaluation --course os --evaluation ew05
####### 24/150 ex3195: John Doe ##################
Graded overall 23 of 150 problems
------- ANSWER -------------------------------------
What would happen if ...
Answer: The following statement would cause ...
grade-or-command> 7
---------- EW05 grades of ex3195: John Doe --------
EW05 grade: 7
EW05 grade log:
       7 ProfA 2022-07-05T14:59:37+03:00
grade-or-command> next


 ####### 25/150 ex3196: Joe Smith ##################
 Graded overall 24 of 150 problems
 ------- ANSWER -------------------------------------
 What would happen if ...
 Answer: The following statement would cause ...
 grade-or-command> 8
 ---------- EW05 grades of ex3196: Joe Smith -------
 EW05 grade: 8
 EW05 grade log:
        8 ProfA 2022-07-05T14:59:48+03:00
 grade-or-command> 9
 ---------- EW05 grades of ex3196: Joe Smith -------
 EW05 grade: 9
 EW05 grade log:
        9 ProfA 2022-07-05T14:59:58+03:00
        8 ProfA 2022-07-05T14:59:48+03:00
```

Figure 5: Automating the grading activity.

The greatest impact is in the evaluation of the final written exam, allowing the grading of 400 students at the same time, by subject, by 10 teachers, in less than 2 hours. In the traditional context of a paper exam, with the same exam structure, it would have taken 6 hours for each of the 10 teachers, plus waiting times for synchronisation among evaluators.

3.2.5 Student Feedback and Status Reporting

The final grade calculation and are done using a custom course-specific algorithm that yields a detailed report of the student activity, which can be emailed to students. In Figure 6 is a truncated sample report. Sa and La are seminar and lab attendances, the grades starting with T refer to quick tests, while those that start with E are practical exam grades. The Ew01, Ew02, ... are the grades for each of the 10 written exam problems, and Ew is the cumulative written exam grade. This detailed report resulted in a significant decrease in the number of contested final grades.

3.3 Practical and Theoretical Student Assessment

Formative evaluation allows the teacher to better explain the objectives and set the expectations, while keeping students engaged and motivated by providing feedback directly and more often. It is used in monitoring, sharing learning goals, and addressing misconceptions (Grover, 2021).

```
Name: John Doe
    Sa = S1 + S2 + S3 + S4 + S5 + S6 + S7 + ...
       =  1 +  1 +  0 +  1 +  1 +  1 +  0 + ...
       =  5
    La = L1 + L2 + L3 + L4 + L5 + L6 + L7 + L8 + ...
       =  1 +  1 +  1 +  1 +  1 +  1 +  1 +  1 + ...
       = 14
    Tc = 1
    Tgsa = 8
    Esh = 8
    Epr = 6
    Eth = 5
    Ew = (Ew01 + Ew02 + Ew03 + Ew04 + Ew05 + ...)/10
       = (10 + 7 + 0 + 9 + 10 + 2 + 10 + 7 + ...)/10
       = 6.9000
    Escore = 1 + 0.3*Ew + 0.2*Esh + 0.15*Epr + ...
           = 1 + 0.3*6.9000 + 0.2*8 + 0.15*6 + ...
           = 7.2700
    Grade = Sa < SaReq || La < LaReq ? Absent
                                     : round(Escore)
          = 5 < 5 || 14 < 12 ? Absent
                             : round(7.2700)
          = 7
```

Figure 6: Computing the final grade - detailed display.

As previously described, in our course planning each chapter ends with a practical lab test. Each student receives a unique account on the server and is randomly assigned a task by the server XM tool to be completed within a time-frame. Students will be involved in doing live programming with command line tools and text editors, and their server sessions are recorded to prevent copy-paste or download activities. The teacher monitors the overall activity during the exam: active sessions, running processes, and IP connections, to prevent remote cheating connections or server overload with processes.

Evaluation of these tasks is performed in the form of a discussion with each student based on the solution they developed for the unique task assigned on the server, randomised by the XM tool. This is performed during the lab classes. Besides grading the final solution, teachers can identify common mistakes, knowledge gaps, and acquired skills, and then discuss them with students, providing clarifications. The grading criteria are very strict, meaning that a code with syntax errors is automatically graded with 1, a program with a severe run-time crash is graded with 4, while 5 is the minimum passing requirement. The maximum grade is 10, and it is obtained if the provided solution correctly addresses the given task in an efficient way (with respect to memory allocation, error checking, and data validation). We also discuss code style, alternative solutions and limitations.

The summative theoretical evaluation consists of a written examination and uses also the exam server, but the tasks are more theoretical. Students are not required a complete program, but a written response related to a specific code or OS context, while still using the skills developed (command line, text editor).

4 SYSTEM EVALUATION AND DISCUSSION

RQ1: Does XM Simplify Grading and Attendance Management for the Teacher?

In order to assess the effectiveness of the XM tool in providing a more efficient management of teaching related activities, the people involved in teaching the OS lecture, seminar and laboratory activities during the analysed period were required to answer a questionnaire, as described in Table 2. The survey contained additional fields on the characteristics of the respondents, such as gender and age. It is important to note that the XM tool is designed for OS teachers specifically, who are proficient in the use of command-line tools, and the assessment of usability is performed from this perspective.

Table 2: Teacher survey questions.

Item	Question
1	In which year(s) did you teach OS?
2	To what extent did the XM tool reduce the time you spent for attendance taking?
3	How did you perform attendance previously?
4	To what extent did the XM tool reduce the time you spent for grading the written exam?
5	How did you perform grading previously?
6	How do you appreciate the system usability?
7	State your reason for the usability assessment.

Questionnaire items are mostly closed questions. Item 1 is a multiple selection from a list of years (the 2016-2022 range). Items 2 and 4 require a single-choice answer from a list of 4 options: None, 10%-25%, 25%-50%, 50%-75%. Item 6 is a single choice of a five-scale quality assessment: Excellent, Good, Average, Poor, Very Poor. Items 3, 5 and 7 are open questions to provide a reference point for the corresponding previous closed questions.

Most respondents are male, aged 30-49, and have been involved in teaching the OS course, seminar or laboratory over several years, before and after the introduction of the XM tool in 2020. In terms of assessing the reduction in time spent with attendance taking, 44% of the respondents estimate a reduction with 25%-50%, while 56% of them estimate a reduction with 50%-70%. The alternative methods for attendance taking used as comparison include: attendance paper manually transcribed in a spread sheet, attendance sheet/electronic spreadsheet, or manually in predefined lists (students signed their attendance on paper then a spreadsheet is completed).

Regarding the impact of the XM tool in reducing time spent with grading, 89% of teachers chose a time reduction of 50%-70% , whereas 11% of respondents chose a reduction of 25%-50%. The previous grading methods for the written exam involve: paper exam grading (sometimes sorting the papers by subject), and the use of spreadsheets.

The usability of the XM tool is appreciated to be Excellent by 89% of the respondents and Good by 11%. The respondents considering that the XM tool interface is intuitive, easy to learn and use, similar to other command line tools. It is also appreciated because it facilitates cheating detection, speeds up attendance tracking and grading, and provides greater fidelity in the correctness of assessments.

In conclusion, XM is an easy to use command line tool that greatly simplifies the class management activities involving attendance and grading by reducing the time required, as opposed to classical approaches.

RQ2: Does Our Approach Facilitate the Learning of Practical Skills for Industry by Students?

To evaluate the impact of this course as it is now structured and how students are receiving it, we have analysed the grades and collected student feedback.

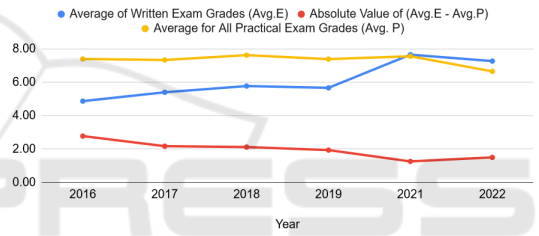

Figure 7: Evolution of the average written exam grade compared to the average of all practical exams grades.

We have analysed the grades between 2016 and 2022, to observe their evolution as teaching approach and tools changed. The year 2020 was omitted from these statistics, since due to the onset of Covid-19, the examination structure was significantly changed from the norm. We compared the grades obtained at the written examination and the grades obtained at the practical examinations (i.e. a comparison of theoretical knowledge vs. practical skills/knowledge). The results are presented in Figure 7 and Table 3.

We also studied the correlation between the grades of the written exam and the average of each practical exam for each student per year (Figure 8 presents the analysis for year 2022). In years 2016, 2018 and 2021 we have observed low values for R^2: 0.232, 0.229 and 0.223, respectively. In 2021, due to COVID-19, all exams were held remote and online, and there was an increase in the number of grades between 8 and 10, potentially skewing the analysis. For years 2016 and 2018 the primary explanation is that there was a discrepancy in the difficulty of the practical examinations and the written examination. In the other years, 2017, 2019 and 2022, with R^2 values of 0.406, 0.325

and 0.3290 respectively, there is a better correlation between the practical and theoretical examinations. This indicates that the evaluation methods that were employed gauge the level of understanding more accurately than in other years.

Table 3: Average of Written Exam Grades (Avg.E) and Practical Exam Grades (Avg.P).

Year	Avg.E	Abs(Avg.E-Avg.P)	Avg. P
2016	4.87	2.78	7.40
2017	5.40	2.18	7.34
2018	5.78	2.12	7.62
2019	5.67	1.94	7.39
2021	7.65	1.27	7.56
2022	7.27	1.51	6.65

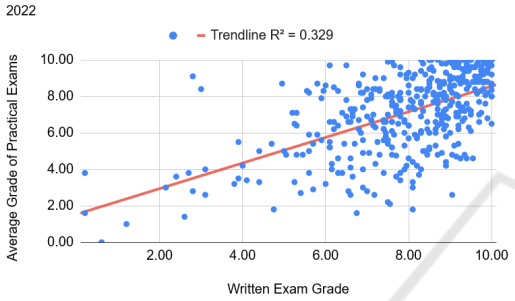

Figure 8: Correlation between theoretical exam grade and average of practical exams grade in year 2022.

We have collected two types of feedback: (a) at the lab and seminar in 2021 and 2022 using Microsoft Forms, with a response of 20% of the students, and (b) the overall course feedback collected in 2022 with 53 responses (meaning 13% of total enrolled students), rating the overall experience of Very Good (38), Good (14) and Bad (1).

Regarding the open questions from both (a) and (b), we have analysed the feedback and grouped similar responses. Although some experiences differ in terms of how labs have been performed (”Teaching method was very interactive”, ”I did not like there was difference in difficulty of lab tests depending on each teacher”, ”So i liked the modern overall approach: everything was done in class no homework and interactive seminars and labs”, ”Teachers happy to explain and interact during the lab”, ”tests graded on the spot”). It is almost unanimous that the students found tests hard and, for that reason, an important motivator to learn and stay engaged throughout the semester: ”The grading system ensured you could pass as long as you had some practice”, ”The tests really stimulated me to learn”, ”Having multiple exams/tests meant there was less material to study for the written exam”, ”The topics at the end of the course are too theoretical”, ”The evaluations were useful because it pressured me to not leave all the material for the exam session”. Most students have stated that they find the subject interesting, that they have a deeper understanding of operating systems, and that they feel that they will definitely use the concepts they have learned later in their career: ”Linux is cool, and I feel like I’ll always use the things I learned during this course in some form or another!”, ”I liked a lot of the things we studied, like how servers work, how memory is managed, how to program in Linux”.

In conclusion, the course method gave students a challenge in a setting they thought would be helpful for their future careers while also involving them in the acquisition of practical skills.

RQ3: What Is the Impact of Our Cheating Detection Mechanism?

One crucial component of the toolkit set-up was cheating detection, which is based on session recordings that can be checked for copy-and-paste-style actions. The total number of such cases identified (graded with 0) for the five practical tests conducted throughout the year are: 119 in 2021 and 153 in 2022. In Figure 9, we have analysed the percentage of detected cheating instances each year, before and after the introduction of our monitoring tool in 2020.

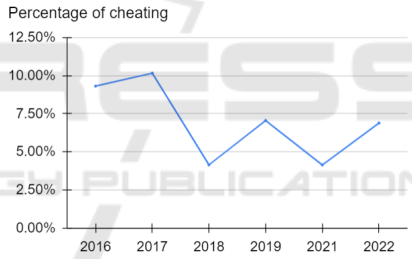

Figure 9: Percentage of cheating for each academic year, calculated as no. of grades 0 from the total no. of grades.

We can observe in Figure 9 that the percentage of cheating that has been identified has decreased starting 2018. We note that there is no visible difference after adding our new approach to detect unauthorised copy-paste actions, whereas previously this was done solely through observations and students’ ability to explain their answer.

In conclusion, our approach can detect cheating that involves copy-paste of code from external sources. The percentage of cheating students is not influenced by the introduction of such a system.

5 CONCLUSIONS

In this article, we present a novel approach and several software tools designed to create an interactive and practical laboratory experience for students tak-

ing the undergraduate Operating Systems course. The XM tool is a command line programme designed to facilitate teaching activities for both teachers and students. The methodology has been put into practice and refined during the last five academic years, with a focus on developing professional and soft skills. We have presented the tools and their role in keeping students engaged throughout the semester by providing ongoing feedback for each practical test.

The outcomes of this experience from both the teacher's and students' perspectives show increased satisfaction and confidence in students' ability to use the concepts learnt forward, in industry. From the teacher's perspective, the system provides more efficient class management (in terms of attendance and grading). The objective evaluation also shows the high-quality practical skills achieved by the students. Moreover, with time and adjustments each year, there is a better correlation between practical and theoretical knowledge and skills, as shown by the higher correlation of grades obtained during the semester and at the end-of-term written exam. However, this needs to be closely monitored and fine-tuned in the future. We note also the system's capability to detect cheating, and this knowledge discourages students, to some extent, from attempting copy-paste of external code.

As future work we aim improvements to the XM software related to plagiarism detection, and to integrate activity patterns detection using artificial intelligence on usage data, to identify drop-out risk.

ACKNOWLEDGEMENTS

The publication of this article was supported by the 2023 Development Fund of the UBB.

REFERENCES

Bailey, J. and Zilles, C. (2019). Uassign: Scalable interactive activities for teaching the unix terminal. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 70–76, New York, NY, USA. Association for Computing Machinery.

Fowler, M. and Zilles, C. (2021). Superficial code-guise: Investigating the impact of surface feature changes on students' programming question scores. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 3–9, New York, NY, USA. Association for Computing Machinery.

Grover, S. (2021). Toward a framework for formative assessment of conceptual learning in k-12 computer science classrooms. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 31–37, New York, NY, USA. Association for Computing Machinery.

Hassan, I. (2020a). Automated authentic assessment: Applied to an undergraduate course in network and system administration. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 183–191, Porto, Portugal. IEEE.

Hassan, I. (2020b). Teaching cybersecurity to computer science students utilizing terminal sessions recording software as a pedagogical tool. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, Uppsala, Sweden. IEEE.

Kankuzi, B. (2019). Balancing theory and practice in an introductory operating systems course. In *ICT Education: 47th Annual Conference of the Southern African Computer Lecturers' Association, SACLA 2018, Gordon's Bay, South Africa, June 18–20, 2018, Revised Selected Papers 47*, pages 362–375. Springer.

Kulik, M. (2022). Asciinema. last accessed on August 8, 2022.

Nieh, J. and Vaill, C. (2006). Experiences teaching operating systems using virtual platforms and linux. *SIGOPS Oper. Syst. Rev.*, 40(2):100–104.

O'Brien, D. (2017). Teaching operating systems concepts with systemtap. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 335–340, New York, NY, USA. Association for Computing Machinery.

Raymond, E. S. (2003). *The art of Unix programming*. Addison-Wesley Professional, Boston.

Sharrock, R., Collin, E., Labat, T., Hamonic, E., Bonfert-Taylor, P., and Goudzwaard, M. (2022). Teaching and learning programming with linux using in-browser client-side web technologies: Exploring the key features for achieving systems and tools scalability. In *Proceedings of the Ninth ACM Conference on Learning @ Scale*, L@S '22, page 427–430, New York, NY, USA. Association for Computing Machinery.

Thompson, L., Clarke, J., and Sheehan, R. (2020). Edufuse a visualizer for user-space file systems. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 549–550, New York, NY, USA. Association for Computing Machinery.

Vavrecková, S. (2020). The combination of skills training for it administrators and programmers. In *ISSEP (CEURWS Volume)*, pages 152–159.

Vykopal, J., Švábenskỳ, V., Seda, P., and Čeleda, P. (2022). Preventing cheating in hands-on lab assignments. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 78–84, New York, NY, USA. Association for Computing Machinery.

Weina, F., Ping, Y., and Shuai, L. (2018). Research on the related teaching method of computer operating system course. In *International Conference on E-Learning, E-Education, and Online Training*, pages 292–297, Cham. Springer, Springer International Publishing.