Introducing Tablet-Based On-Site E-exams in a Large Software Development Course: An Experience Report

Paula Rachow[1], Christian Rahe[2], André van Hoorn[3]

Abstract: For introductory programming and software development courses, an electronic examination format with programming tasks in an IDE-like environment seems like the natural choice. However, for our courses at the Universität Hamburg, with up to 450 exam participants, we have so far still conducted paper-based exams that are time-consuming to grade. In the context of modernizing the assessment methods, this paper explores and reflects on introducing tablet-based e-exams in the introductory module "Software Development 2". This experience report highlights the motivation, challenges, setup, outcomes, and lessons learned of this transition. The implementation of e-exams mirrors industry practices and aligns with the course's learning objectives. Moreover, the results showcase improved efficiency in grading, reduced logistical challenges, and enhanced accessibility. However, adopting e-exams also introduces challenges, including a dependency on a robust technological infrastructure. This paper offers valuable insights into the advantages and risks for educators seeking to integrate e-exams into their curriculum. Despite initial challenges, the positive outcomes of this transition encourage us to continue integrating e-exams into our curriculum in the future.

Keywords: Digital e-exams; Programming education; e-Learning; Automated grading

1 Introduction

In the summer semester of 2023, we significantly changed how we assess students in the final exams in the foundational programming module "Software Development 2" at the Universität Hamburg. We shifted from using traditional paper-based exams to on-site tablet-based e-exams with the learning management system Moodle[4] for the summative assessment. We were driven by our desire to make assessments more practical and efficient. Since we were already familiar with using Moodle and had learned from our experience with remote e-exams in 2021 (as a response to the COVID-19 pandemic), we felt prepared for this change. This paper aims to share the reasons behind this shift, explains how we adapted to our specific situation, describes the technology we used, and passes on our learned lessons. We hope to guide other educators who may be considering a similar transition.

[1] Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Deutschland, paula.rachow@uni-hamburg.de
[2] Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Deutschland, christian.rahe@uni-hamburg.de
[3] Universität Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Deutschland, andre.van.hoorn@uni-hamburg.de
[4] moodle.org

doi:10.18420/seuh2024_03

2 Background: Digital assessment

E-exams are digital exams conducted through an e-testing system [Ba21] and they can be held either on-site or remotely. They come in two formats: open-book and closed-book. Open-book e-exams allow students access to reference materials, encouraging them to synthesize information and apply critical thinking. Closed-book e-exams require students to rely solely on their knowledge, skill, and their comprehension of the subject matter.

The whitepaper by Bandelt et al. [Ba21] lists some practical examples of digital exam implementations at different universities, including e-exams. For instance, the University of Zurich shares similarities with our approach, but they rely on a large computer pool, and do not employ tablets as we do [Ha14]. The University of Applied Science Munich provides the automation framework *EXaHM* for digital exams to steer the exam process, like the startup and shutdown of the PCs[5]. Other papers about digital exams often deal with other scenarios, e.g., Bottcher et al. [BT23] report about conducting remote open book exams.

More study was devoted to learning management systems and online programming platforms utilized throughout the semester, such as Artemis [KS18] and OPPSEE [SB23]. However, there is a notable gap in the literature addressing their specific role in exams. Aspects such as automated feedback [Be21] and strategies for more effective manual grading of student exercises [La19] are crucial but are less relevant in the context of e-exams. Gandraß and Schmolitzky [GS19] present another Moodle plugin for programming exercises that looks similar to the CodeRunner plugin used in our setting and described in a later section.

3 Our drivers for switching to an e-exam

For us, a strong motivation for implementing e-exams in the context of the course included several compelling expectations.

Real-world relevance. E-exams provide a practical assessment environment mirroring real-world software development, where digital tools like compilers and IDEs are commonplace. By providing the students with a compiler for coding tasks, students are assessed in a more realistic environment. It allows them to concentrate on understanding and applying core programming concepts rather than memorizing syntax.

Constructive alignment. Unlike traditional paper-based assessments, e-exams closely resemble the learning objectives and the exercise contents. They emphasize comprehension and application, discouraging mere memorization. This tailored approach ensures that the evaluation accurately measures whether students have acquired programming skills.

Fast feedback. Grading of e-exams promises a high degree of automation, allowing students to receive timely feedback on their performance, ensuring that students who failed have enough time to prepare for the second exam date in the same semester.

[5] `https://www.hm.edu/lehren/kompetenzzentrum_digitales_pruefen/EXaHM.de.html`

Reduced effort. Handling physical exam papers, ensuring fairness in grading, and managing logistical challenges are time-consuming for educators. E-exams can streamline this process, enabling instructors to focus more on teaching and content development.

Error mitigation. With e-exams, there are fewer sources for errors while calculating points, and grading the exams as this is done automatically. Furthermore, changes to the accepted solutions or grading scheme can be automatically applied to all submissions.

Customized question types. Electronic platforms offer diverse question types beyond multiple-choice, especially coding exercises, allowing for a more comprehensive evaluation of the students' practical programming skills.

Accessibility and inclusivity. E-exams support diverse needs with features like adjustable fonts, screen readers, and alternative input and, therefore, promote inclusivity and easy accommodations for students with approved reasonable (special) accommodations.

Environmentally friendly. E-exams reduce the need for physical paper, promoting environmental sustainability by reducing waste.

Data-driven insights. Electronic platforms often generate analytics and reports, providing educators with valuable data on student performance trends, areas of struggle, and overall class performance.

Economic. E-exams eliminate the need for printing — cutting costs on paper, ink, and equipment. There is also less need for physical storage space afterward and no student assistants have to be employed to help with the grading.

4 Our setting

4.1 Course context

This paper presents a reflection on introducing on-site closed-book e-exams for the "Software Development 2" module at the Universität Hamburg. The module is attended by approximately 450 students each summer semester. The students, typically in their second semester, are enrolled in various bachelor programs, including informatics, human-computer interaction, computing in science, and information systems.

The module provides students with a foundational understanding of software development principles and practices — with a focus on object-orientation. It comprises one weekly lecture and a two-hour in-person exercise session. During these sessions, the students work on programming exercises in pairs or (in the second part of the semester) in groups of four. In addition to these sessions, students complete mandatory weekly Moodle quizzes — including programming tasks — to reinforce their learning.

There are two date options to take the exam each summer semester. The exam has a duration

of 120 minutes. In the summer semester 2023, 203 students participated on the first exam date and 108 students participated on the second exam date.

4.2 Lessons learned from a prior remote e-exam

During the COVID-19 pandemic, we had already implemented an e-exam in 2021. However, this e-exam was conducted remotely and in an open-book format. Through this experience, we gained a pool of e-exam questions and identified several critical issues that have informed our decision-making process for the transition to an on-site closed-book e-exam.

Scalability. We encountered significant scalability issues with our Moodle platform — particularly related to the CodeRunner plugin and back-end. After all the exams were automatically submitted and the results were calculated, the system experienced a critical breakdown. Luckily, only the student changes from the last few seconds were lost. This incident highlighted the need for a more robust and scalable infrastructure to support the increasing demand for digital assessments. However, it also showed that an e-exam with many students is possible.

Foundational course. Our module is a foundational course in the second semester. According to Wollersheim et al. [WMS11], foundational courses primarily target the lower levels of cognitive learning, which encompass 'knowing' and 'understanding' [Kr75]. Therefore, crafting questions for an open-book exam posed a challenge. It is essential to balance that questions are not easily searchable online and still have a reasonable level of complexity. It convinced us to perform only closed-book tests in the future.

Cheating. We encountered multiple instances of cheating. Therefore, we were concerned about the integrity of remote exams, as it is difficult to ascertain who is in front of the computer and whether any unauthorized assistance is present in the room. Especially since the Universität Hamburg does not allow online proctoring of students, which further complicates the issue. Due to these reasons, we preferred to stay with on-site exams.

Communication. We realized that direct communication on-site with students is notably easier, especially for delivering announcements and addressing last-minute concerns or queries. It also simplifies communication among exam supervisors, allowing for seamless coordination and immediate response to unforeseen circumstances.

Responsibility shift. A take-home exam shifts the responsibility to students for a stable internet connection and suitable testing conditions, potentially disadvantaging some. Despite the tablet being less ergonomically comfortable than the students' usual setups, on-site e-exams with tablets provide a controlled environment and are a good option when a large pool of computers does not exist.

4.3 Procedure and effort comparison

In this section, we compare procedural aspects and effort involved in conducting an e-exam instead of a traditional paper-based exam. Table 1 summarizes the processes/tasks and (estimated) efforts elaborated below. By examining each process step, we aim to provide a detailed insight into the logistical considerations and administrative workload associated with both assessment methods. The shift to e-exams alters time allocation, with more even planning across the semester. During the exam, there are additional technical questions but after the exam, tasks like grading happen faster. Additionally, a significant portion of the additional effort is redirected toward the IT administration.

Tab. 1: Effort of processes and tasks for e-exams compared to paper-based exams.

Task	**1st iteration**	**Future iterations**
Preparation	++	+
Creation of exams	++	+
Organization	+	o
Execution	+	+
Supervision	+	+
Post-processing	-	- -
Grading	-	- -
Analysis	- -	- -
Exam review	-	- -

- - significantly less effort, - less effort, o same effort, + more effort, ++ significantly more effort

4.3.1 Preparation

Creating exam questions is not more complex than with a paper-based exam in LaTeX. While Moodle allows for more complex code tasks, the assurance of the automatic evaluation induces more effort during setup, in addition to the effort of the course creation, configuration settings, and student enrollment. However, the organization of the printing is no longer necessary. In addition, the tasks can be randomized and random questions can be drawn so that the degree of reuse of the exam tasks is high.

Managing tablets — updating software, charging batteries, and preparing the e-exam configuration — also entails a higher overhead. However, in our case, the responsibility for the tablets lies with the IT administration staff.

4.3.2 Execution

We had additional technical assistance from the IT administration team to ensure a smooth process. They provided technical support and facilitated the delivery and distribution of the tablets. For us, the necessity of picking up and carrying paper exams has been eliminated.

Our university only has 210 tablets (iPads) available for e-exams. Therefore, we used additional PC pools. In the future, we plan to extend the lecture hall reservation and let the students write in two cohorts.

4.3.3 Post-processing

Exam grading. With the paper-based exam, we had to organize the exam grading process, including finding and preparing rooms, graders, and files. The exam grading used to take up to 4 days with two instructors and multiple student assistants. The grading per exam took approximately 35 minutes including an additional review of each grading. Grades were usually published two weeks after the exam. With the e-exam, the grading is done automatically and the publication on the same day.

Exam review. For the exam review, we had to look for the students' exams and hand them out. With Moodle, we can just change the permission for the student to be able to see the results. There is also less chance that the grading has to be changed since the grading is performed automatically. Moreover, if we notice a general error in the rules for the automated grading, we can easily reassess all exams automatically.

5 Technical realization of the e-exam

Fig. 1: Students taking e-exam.

Fig. 2: The iPad setup with keyboard.

5.1 Moodle setup

We implemented our e-exam in Moodle, a popular open-source learning management system. We chose Moodle because the MIN faculty at the Universität Hamburg operates its own instance and we and the students were already familiar with using it during the semester. Each e-exam has a dedicated Moodle course, implemented as a Moodle *test* activity with password protection. Each student has an individual timer of 120 minutes, after which the test terminates automatically. The timer can be extended for groups or individual students, especially for those with approved special accommodations, granting them the necessary extra time for the exam. The setting "Full screen pop-up with some JavaScript security" ensures an undistracted exam environment by hiding chat and menus.

5.2 Hardware setup

Students used 9th generation iPads with integrated keyboards for the exam (see Fig. 2) as they are portable and easy to manage. Managed through the Jamf School fully supervised MDM system[6], these iPads offer precise control over functionalities, with deliberate restrictions on features like the microphone, camera, and auto-fill options. Bluetooth is deactivated to prevent external interferences and the iPad exclusively connects to the eduroam network. Strict DNS filtering controls access to specific domains, allowing connections only to designated university servers. The exam link is easily accessible through a home screen icon (WebClip), ensuring quick deployment. For future reference, the tablet's device ID is transmitted during the exam. To maintain a secure testing environment, browser navigation is disabled during the exam. For the computers, we used the Safe Exam Browser[7] to limit access exclusively to the e-exam.

5.3 Question types

We implemented a Moodle test with 33 to 37 questions per exam, summing up to 120 points. The exam takes 120 minutes so the students can estimate that they should take one minute per point. These exams closely mirrored the format of previous paper-based exams. The questions were mostly independent of each other and covered the following question types:

Single choice. We implemented single-choice questions through the multiple-choice question type in Moodle. E.g., we quizzed the students about the definitions of the SOLID principles.

Drop down. We implemented various drop-down questions: Cloze (gap fill) questions, selectable short answers, and matching questions. Examples are a cloze for the design-by-contract model and matching Java streams to their corresponding for-loop versions.

CodeRunner. The CodeRunner[8] is a plugin for Moodle and was already provided by the university. Unlike traditional formats, CodeRunner requires students to write and execute code. By emphasizing the execution of code over memorizing syntax, CodeRunner aligns with the module's emphasis on practical application and critical thinking in software development. Additionally, assessments in the CodeRunner format are equipped with a compiler, allowing students to concentrate on algorithmic logic and problem-solving strategies without becoming entangled in syntactical details. This format proves invaluable in preparing students for the coding challenges they may encounter in their future careers. In addition, we implemented a graphical UML tool with the CodeRunner plugin to test the students' UML knowledge. Fig. 3 shows a CodeRunner example question where the students had to rewrite the code with a lambda expression.

[6] https://www.jamfschool.com/

[7] `https://safeexambrowser.org`

[8] `https://moodle.org/plugins/qtype_coderunner`

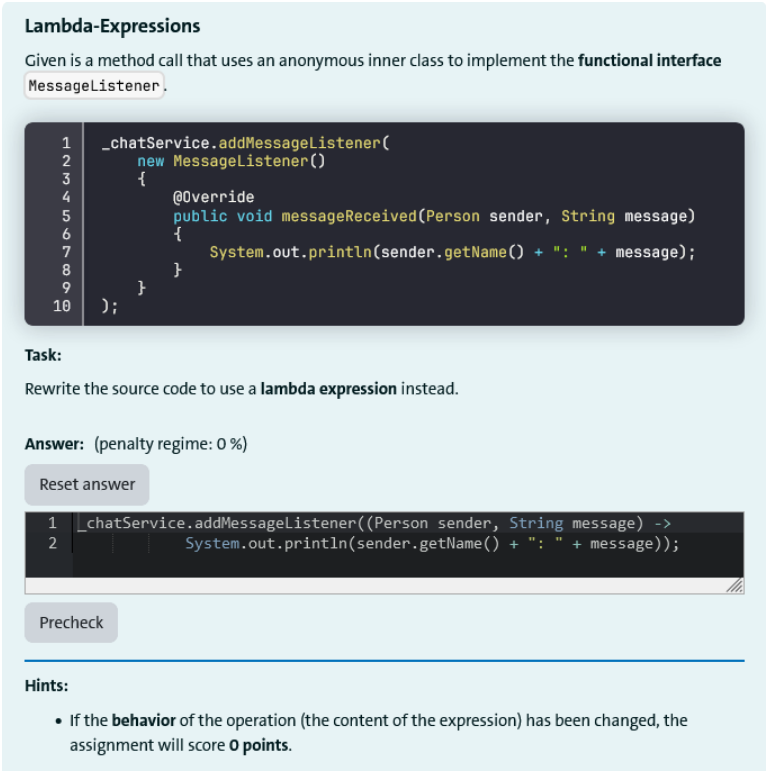

Fig. 3: A programming question about lambda expressions.

Excluded question types. We excluded question types like drag-and-drop because they were not consistently working on the tablet. We also excluded the "Random Short-Answer Matching question type" in the second exam because there was a bug with the evaluation[9]. Moreover, we did not use free text or similar question types because we wanted to be able to evaluate everything automatically.

5.4 Moodle extensions

We implemented minor UI enhancements using a Moodle text block in the exam course's sidebar, containing a *script* tag and incorporating browser-side JavaScript for page modification. One such improvement involved displaying each student's name and ID number at the top of the browser window. This streamlined the identity verification process for exam supervisors, eliminating the need for students to navigate or scroll to their profile, thereby

[9] https://tracker.moodle.org/browse/MDL-76676

expediting the process. Additionally, we modified the submission timer in Moodle to only display seconds during the last ten minutes, reducing visual distractions.

When a quiz is submitted in Moodle, all answers, including CodeRunner questions with potentially resource-intensive testing logic, are automatically processed for grading. Currently, there is no option to disable this feature. Recognizing the potential strain on our infrastructure from a surge of simultaneous submissions, we took proactive measures to prevent exams from being submitted manually in the first place.

Preventing auto-submissions. We configured the exam test activity with the timing option "Attempts must be submitted before time expires, or they are not counted". Using JavaScript injection, we prevented the timer from expiring automatically, thus avoiding premature grading. This way, for students with an expiring timer, no automatic grading was initiated. Students were informed in advance that their exam status would show as "Never submitted".

Preventing manual submissions. In our exam setup, students were able to exit the exam early. To prevent unintended automatic grading while others were still working, we disabled the "Submit" button. Instead, we instructed students to seek a supervisor's assistance for submission. The supervisor would then apply a user override to conclude the exam, resulting in the "Never submitted" status and averting automatic grading.

5.5 CodeRunner extensions

The Java template in CodeRunner combines student answer code and test code in a single compilation unit. If the student code doesn't compile or the test code accesses missing elements, the raw compilation error is presented. This can be confusing for students due to line number offsets and errors referencing code they have neither written nor seen before.

Missing methods. In exams, we consider it essential to award part of a question's points for incomplete answers wherever possible. To enable testing of student code with only a subset of required methods, we utilized the Java Reflection API for property accesses, constructor, and method calls. This approach allows specific test cases to fail with a runtime exception, rather than rendering the entire answer invalid.

Compilation errors. For the majority of code questions, compilation was required to receive credit. In some cases, however, we wanted to recognize partially correct answers even if they had syntax errors. To ensure we could still run test code if the student code did not compile, we inserted the student submission into our CodeRunner test template as a string and compiled it in memory using the *JavaCompiler* class. This approach allowed us to analyze properties of the submitted source code, e.g. to award partial credit for the presence of certain methods or constructs, even if we could not test its runtime behavior.

Introducing an additional computational cost per question granted us valuable flexibility in question design. For instance, in one question, students needed to identify and fix three

separate compilation errors in a provided code snippet. We were able to award partial credit even if errors persisted in the student's code.

6 Lessons learned

In this section, we explore lessons learned from implementing e-exams, covering general and technical insights, each rooted in one of the expectations that led to this transition.

Real-world relevance: We were able to pose more complex coding tasks and the students were able to use a compiler to iteratively solve them. An IDE would have been even better and closer to practical programming, but it also comes with additional effort.

Constructive alignment: The students were already familiar with the type of questions and the environment from the Moodle quizzes during the semester and could focus on the questions rather than the technical infrastructure.

Fast feedback: If there are no complications in grading — such as server overload or solutions — the grades can be published on the same day.

Reduced effort: Efficiency improved significantly with reduced grading efforts. However, there is an increased effort for the initial question creation, mainly because we wanted to be able to grade each question automatically. However, we hope that the incorporation of randomization rendered questions highly reusable. For the coding questions, the students' answers usually must be compilable for us to grade them automatically. While this poses a slight disadvantage for students with partially correct answers, we believe it remains feasible and fair, thanks to the compiler's availability and unlimited compiler runs for the students.

There was a considerable demand for IT system administration. To further optimize the process, dedicated exam rooms with proper technological infrastructure could mitigate technical issues and provide a controlled, focused examination environment.

Error mitigation: We enhanced accuracy by eliminating error-prone steps, including manual grading, manual data calculation, and manual transfer to Excel and Stine[10], the information system of the Universität Hamburg. This transition to automated grading significantly enhanced accuracy in the assessment process. This also became obvious when there were no grade corrections after the exam review and no significant changes in the automatic assessment.

Customized question types: Moodle questions offer students a more intuitive answering experience by making the desired type of answer clearer. Moodle questions are also easier for instructors to create compared to paper-based exams in LaTeX, which can be more time-consuming and may not provide the same level of customization.

[10] https://www.stine.uni-hamburg.de/

Accessibility and inclusivity: While we acknowledge the advantages, we did not need it for the initial two e-exam instances.

Environmentally friendly: For the previous exams, we had to print 10,500 pages for the two exams that are now no longer necessary.

Data-driven insights: Moodle automatically provides the facility index and discriminative efficiency for each question[11]. It also indicates which questions need a review depending on the grading. Currently, the grade distribution appears to be absent and there is only a point distribution diagram in increments of ten points.

Economic: There was no need to pay any student assistants to help with the grading.

7 Risks and Constraints

Despite the many advantages of e-exams, there are also additional risks and constraints:

Strong dependency on infrastructure. Successful e-exams rely heavily on technological components such as tablets, Wi-Fi, platforms like Moodle, and authentication services, all requiring reliable operation. In large organizations like universities, avoiding maintenance during exams poses a challenge. In our case, a pre-scheduled and non-reschedulable maintenance window coincided with the first exam. Fortunately, all systems operated smoothly on the day. Going forward, we have proactively planned around maintenance dates to preempt any potential disruptions to the examination process.

Limited infrastructure capacity (scalability): Implementing e-exams places a significant demand on the infrastructure. The system must be robust enough to handle a large number of students simultaneously accessing resources, submitting responses, and interacting with the assessment platform. Scaling up the infrastructure to accommodate a surge in usage during exam periods is essential to prevent potential technical bottlenecks.

Capacity of devices, rooms, and resources: With only 210 available tablets, accommodating more students necessitates a sequential exam approach which takes more time and can potentially lead to scheduling conflicts. This year, we had to let some students participate in the PC pool because we could not extend the lecture hall reservation. For the future, we booked double the amount of time to allow for two cohorts.

Additional error sources: In e-exams, potential errors can stem from technical settings and human factors. For instance, incorrect configurations or platform settings can disrupt the exam process. Additionally, students may face challenges with passwords, leading to delays. To ensure a smooth assessment, educators must carefully set up the digital exam environment and have plans to address any unforeseen issues swiftly.

[11] https://docs.moodle.org/403/en/Quiz_statistics_report

8 Conclusion

In conclusion, introducing e-exams in our software development course has proven highly beneficial. Positive student feedback reaffirms the effectiveness of this transition aligning with the findings of Martin et al. [Ma22] that digital formats reduce stress and anxiety. Notably, we observed no significant differences in grade distribution compared to previous paper-based exams. By entrusting the technical tasks and responsibilities to the dedicated team at the university, we have significantly reduced our administrative workload. This shift allows us to concentrate our efforts on the core aspects of teaching and content refinement. Looking ahead, we anticipate that the long-term benefits will outweigh the initial implementation expenses.

References

[Ba21] Bandtel, Matthias et al.: Digitale Prüfungen in der Hochschule. Hochschulforum Digitalisierung, 2021.

[Be21] Bernius, Jan Philip et al.: A machine learning approach for suggesting feedback in textual exercises in large courses. In: Proc. 8th L-AT-S. 2021.

[BT23] Böttcher, A.; Thurner, V.: Digitale Prüfungen für Softwareentwicklung im "Bring Your Own Device, Open Book, Open Web"-Format. In: SEUH 2023. 2023.

[GS19] Gandraß, N.; Schmolitzky, A.: Automatisierte Bewertung von Java-Programmieraufgaben im Rahmen einer Moodle E-Learning Plattform. In: Proc. ABP. 2019.

[Ha14] Halbherr, Tobias et al.: Making examinations more valid, meaningful and motivating: The online exams service at ETH Zurich. In: EUNIS Journal of Higher Education. 2014.

[Kr75] Krathwohl, David R et al.: Taxonomie von Lernzielen im affektiven Bereich. Beltz, 1975.

[KS18] Krusche, S.; Seitz, A.: Artemis: An automatic assessment management system for interactive learning. In: Proc. 49th ACM SIGCSE. 2018.

[La19] Laß, Christopher et al.: Stager: Simplifying the Manual Assessment of Programming Exercises. In: SEUH 2019. 2019.

[Ma22] Martin, Robert J et al.: Digital, Online, Take-Home–University Students' Attitude towards Different Examination Formats. In: IEEE GeCon. 2022.

[SB23] Schmolitzky, A.; Burau, H.: OPPSEE-Eine Online-Plattform zum Programmieren Üben. In: SEUH 2023. 2023.

[WMS11] Wollersheim, H.-W.; März, M.; Schminder, J.: Digitale Prüfungsformate. Zum Wandel von Prüfungskultur und Prüfungspraxis in modularisierten Studiengängen. Zeitschrift für Pädagogik 57/3, 2011.