Rate Your Mate for Food for Thought: Elsewhere Use a Grader

Pia Niemelä and Mikko Nurminen
Tampere Universities, PO Box 527, FI-33101, Tampere, Finland

Keywords: Tools to Assess Learning, Instructional Design, Mentoring and Tutoring, Metrics and Performance Measurement, Project based Learning, Engineering Education, Flipped Classroom.

Abstract: Finnish university pedagogues are keen on applying flipped learning techniques to improve education and learning outcomes. Flipped learning implies the transfer of assessment in a more formative direction and targeted feedback that is frequently delivered. On the contrary, teaching resources are decreasing. Increasing the portion of self-, peer- and automatic assessment partially helps solving this dilemma. Currently, Tampere University is in the midst of the process of combining two separate campuses together. Both campuses offer major-specific computer science studies. This paper presents a case study of merging basic level web technology courses, and in particular their different assessment practices together. The courses are targeted for the second- and third-year students, and the number of participants is about 200 in the studied course (enrolled N=324 / completed N=178). The merged course was run in two learning management systems (LMSs), called Plussa and WETO. The switch from one LMS to another happened in the middle of the course. LMSs employed different assessment practices for weekly exercises: Plussa assessed them automatically, whereas WETO exploited peer-reviews. This provided a unique opportunity to compare these two assessment methods and the study addresses the pedagogical opportunities and challenges of both.

1 INTRODUCTION

Flipped learning and assessment prepare students for the 21st century, where the requirements of new flexible working life require continuous learning and self-development, accelerated by self-reflective practices. Flipped classroom and flipped learning are trending with the promise of improved and intensified learning (Graziano, 2017; Akçayır and Akçayır, 2018), and self-learning in particular (Tan et al., 2017).

This paper describes the steps towards flipped learning in one introductory computer science course arranged at Tampere University, and examines the assessment practices in particular. The research questions are:

- RQ1: How to ensure an adequate amount of feedback in mass courses with scarce teaching resources?
- RQ2: How do the course participants perceive peer-reviews and automatic assessment?

First, the Related Work section reviews the development of assessment practices in computer science. Next sections, ‘Research context’ and ‘Method and research instruments’ document our research set-up; ‘Results and Discussion’ section reports our findings and generalizes them to recommendations. Conclusions summarize the lessons learned.

2 RELATED WORK

Future work-life demands not only digital skills, but also soft skills, flexibility, and self-regulation (Sennett, 2011; Gupta, 2017; Finch et al., 2013). Flat and agile software teams get assignments, and share and manage tasks independently by taking themselves the care of the project management. The success of these teams depends largely on how skillfully they are able to proceed towards a shared goal (López-Alcarria et al., 2019). For example, Arum and Roksa advocate the importance of authentic projects in preparation for future working life, and advanced socio-epistemic practices for solving complex challenges, promoting innovations, and acting in an ethically responsible way (Arum and Roksa, 2011).

To respond to the requirements of future work, higher education has started to update its practices, e.g., assignments are transformed in an open-ended direction that is characteristic both to problem- (Tandogan and Orhan, 2007) and project-based learning

Niemelä, P. and Nurminen, M.
Rate Your Mate for Food for Thought: Elsewhere Use a Grader.
DOI: 10.5220/0009564204220429
In *Proceedings of the 12th International Conference on Computer Supported Education (CSEDU 2020) - Volume 2*, pages 422-429
ISBN: 978-989-758-417-6

(Vogler et al., 2018). More of the responsibility for learning is moved from the instructor to the learner. A skillful learner is able to vary the learning strategies and techniques by selecting the best methods and means for each learning context referred to as meta-cognitive skills (Joseph, 2009; Pintrich, 2002). Meta-cognitive and problem-solving skills complement each other. In education, these skills can be strengthened by guiding students to function in autonomous Agile teams that plan, control, schedule and self-reflect the progress in Scrum meetings, and continuously improve the deliverables with proper quality measures in place (Gannod et al., 2015).

Moving towards student-directed learning should be reflected in the assessment practices as well. Assessment, especially self-assessment, such as students' self-reflection on their goals and reviewing their own competence is essential in developing a real expertise (Schön, 1991). Therefore, assessment should orient towards the reflective practices of continuous improvement. Flipped learning facilitates learner's autonomy by granting more freedom to students structuring and scheduling their studies, thus directing to the self-regulation (Toivola et al., 2017; Sointu et al., 2019). The latest, the most far-fetched implementation of flipped learning suggests flipping of assessment as well (Toivola, 2020). Flipped assessment implies students' own choices on the challenge level: the more concise the content a student selects – the lower the grade.

In CS education, agile-managed group work exemplifies project-based learning in its purest. Git is used to version the code and share it; its usage is learned in the so-called 'authentic context' (Haaranen and Lehtinen, 2015). In the CS domain, flipped assessment actualizes in test-driven development: students write their own tests, and after each Git commit source code is automatically tested by the continuous integration system (CI). In consequence, students themselves can control the quality with tests and linting, in a flipped manner. Test coverage is an objective measure of the tests themselves; it impartially evaluates the quality of the tests. In this course's setup, Gitlab also provides the CI functionality, thus functioning as one alternative learning management system in addition to Plussa and WETO (Niemelä and Hyyrö, 2019).

3 RESEARCH CONTEXT

The analyzed course is a comprehensive introduction to both frontend and backend web technologies and it is targeted to second-/third-year students. Frontend technologies comprise HTML5, CSS, and JavaScript, whereas backend introduces NodeJS and supporting frameworks for implementations such as Express and Handlebars. (NodeJS replaced previous Python Django as a backend technology.) The hidden curriculum is also to teach Git, agile project management with Gitlab Issue Board, and DevOps (the CI part, CD was not included): skills that are highly appreciated by employees and comply hermetically with project-based learning.

Our application of flipped learning is partial: the course consists of lectures that were video recordings in the first- and live videos/video recordings in the second period; weekly hands-on exercises that were scaffolded; and a coursework assignment done in groups of three as the final exercise. Students struggling with the exercises could get help both in Kooditorio and via Mattermost discussion channel. Kooditorio is a tutoring practice a-kin to primetime (Koskinen et al., 2018), except voluntary, where teachers and assistants answer questions, debug and co-implement students' code and scaffold them finalizing their exercises. In most courses, Kooditorios are provided once a week, ca. 2-3 hours per session. Also reference implementations for weekly exercises may be represented that is up to teachers' own volition.

Mattermost is a discussion forum, comparable to Slack. During the course, it was used as a typical Q&A channel, where primarily students instead of teachers were encouraged to respond to the questions and help each other. Joining Mattermost was a recommended practice yet not mandatory, the recommendation was followed by the majority of the students (N=240). During exercises and coursework assignment, the discussion was lively and the channel was extensively used. Overall, compared with fully-fledged flipped learning, this course can be described as its resource-optimized sibling, where pretests were not as clean-cut (weekly exercises), and primetime not as demanding as in the orthodox versions of flipping since the participation was voluntary.

The assessing and the exercise return system differ remarkably both in WETO and Plussa. WETO asks students to upload separate or zipped files, whereas exercises in Plussa ask for a Git repository URL. Plussa goes together with the self-implemented convenience tool, Repolainen. It creates student- and group repositories in Gitlab. The creation is done at the beginning of the course, and group repositories after group formation. As an input, Repolainen gets a participant or group list and as an output it provides Git repositories, attaching a course personnel as maintainers and students as developers. For others course participants, students' repositories are private. CI/CD pipelines are configured in .gitlab-ci.yml file of a template project, and they are executed by the Gitlab group runner. Weekly exercises were stored in the student's private Git repositories, and the coursework assignment to the group's Git repository. Groups were advised to list and assign tasks in Gitlab Issue Board, which is Kanban-style software project management tool. All in all, if advice was taken, Gitlab provided a panoptic view for the progress of each group.

The group work was kicked-off with a 15-minute introduction session, where course personnel answered the questions and gave group-specific guidance: this was found as a good practice and managed to clarify the targets of the assignment. Kooditorio and Mattermost were provided till the very end of the course as a supplementary support.

The assessment of the coursework assignment was automated as far as possible. Without automation, the amount of work would have been enormous: 66 assignments allocated for one teacher was overkill: the assessment practice was largely dictated by coercion. Scoring weights were stated in the instructions; the goal was to offer as many tests as possible for students. To a large extent, it was achieved: most of the unit tests were provided right from the beginning, and in the middle of the assignment period linting and coverage tools were released.

3.1 Method and Research Instruments

In addition to different LMSs (Plussa and WETO), also the name and duration of the course were different. In Hervanta Campus, the course is called Basic Web Application (BWA), its duration is two periods. In Centre Campus, its name is TIETA12, and duration is one period that overlapped with the latter period of BWA. Students' views about their learning outcomes and different assessment practices were collected in the beginning (pre-test) and in the end (post-test). BWA students filled the questionnaire at the end of the first period; TIETA12 students filled the same questionnaire at the start of the second period. The post-test was common for both student groups at the end of the course.

The post-tests divide in two parts; the first was called Rate-your-mate, the latter Rate-your-learning. Rate-your-mate questions were shortened from the reflection tool 'Team Q' (Britton et al., 2017). This course exploited the questionnaire as an instrument of reflecting each group's socio-epistemic practices and members' group competences. A more prosaic target was to identify freewheelers, where peer-reviews complemented the commit numbers and a distribution within a group. The weight of peer-review rating in the assessment was 10%.

3.1.1 Pre-/Post-test Questionnaire

The questionnaire comprised following questions, and it was filled both before and after the WETO period.

- What is your level of experience with peer-review assessment?
- How well do peer- and self-assessment function in your opinion? If you have not used any peer-review system, here is a short description of WETO: - a random student's work is given to another student (i.e., peer) to assess - a detailed assessment instruction is provided for peer-reviewers to help them to do their job - the peer-reviewers have to assess their own work (self-assessment) and two other works - the course personnel reviews the peer-reviews, this phase is called meta-review.
- How have you benefited from reviews from your peers in any context?
- Which cons do you associate with peer-reviewing?
- Which peer-review systems have you used?
- How would you develop or improve peer-review systems further?

3.1.2 Rate-your-Mate Questions of Post-test

Likert-scale questions in the beginning were for teachers' information only, the free word feedback as the last question was delivered for the reviewee to see. Other members rated their mates based on the following statements in the range from 1 to 5:

- Participates actively and accepts a fair share of the group work
- Works skillfully on assigned tasks and completes them on time
- Gives timely, constructive feedback to team members, in the appropriate format
- Communicates actively and constructively
- Encourages all perspectives be considered and acknowledges contributions of others
- Constructively builds on contributions of others and integrates own work with work of others
- Takes on an appropriate role in group
- Clarifies goals and plans the project
- Reports to the team on progress

The free text feedback was: 'Other positive and/or negative feedback about the collaboration during the group work for this person. This feedback is shown to the receiver.' The team collaboration questionnaire used as a reference (Britton et al., 2017) was more verbose; only the questions relevant for this course's context were selected.

3.2 The Course Assignment and Its Automatic Assessment

In the 2019 course implementation, the grading of coursework assignments combined:

- automatic assessment (0..3)
- usability/user experience evaluation
- peer-reviews within a group

The weights of each component were following:

1. 40% unit tests
2. 20% eslint
3. 10% coverage
4. 10% number of tests created by the groups
5. 10% active Git usage: the number of commits, an even commit flow anticipated; the even commit contribution among group members being a bonus
6. 10% rate-your-mate: peer-reviews among group members, done with Plussa's peer-review component. Originally, the idea was to catch freewheelers, where peer-reviews provided necessary information complemented by Git commits. Its pedagogical value was retrospectively noticed afterwards.

The target was the normal distribution of grades, that is, the final limits were checked only after the results were known for the whole group.

This evaluation grants the points in the range of 0..+3. The only criterion is not to pass the unit and integration tests, but the application must be functional as well, concerning all parts: the game, management view, as well as self-written tests. The course personnel reviews the application and its documentation. In particular, considering security concerns was appreciated.

During the review, the course personnel may grant: +1 if the assignment is exceptionally well done -1, if the opposite, or even decline the work, if all the signs indicate freewheeling or all the essential functionality in the application is missing. In this case, the other group members' results may get compensated. In most of the cases, however, the course personnel will leave the auto-assessed grades intact.

3.2.1 Gitlab as an Analysis and Measurement Tool

In the starting lectures in both campuses, the principle 'commit early - commit often' was emphasized. A steady flow of commits is preferable, and it benefits both the group and the course personnel by providing an improved visibility for follow-up. Gitlab has a versatile and well-functioning REST API to query the results of unit tests from executed CI pipelines. In addition, Python provides libraries (pydriller) for examining locally-cloned Git repositories. Having a proper alarm mechanism in place for Git commit contributions (and contribution anomalies) would help early interventions with groups in case of problems, and the early detection of uneven distribution of work.

4 RESULTS AND DISCUSSION

This section first dissects the self-reported learning outcome, then the course grades and assessment results.

4.1 Self-reported Knowledge of Syllabus Topics (Pre- and Post-test)

CS students scored their knowledge on the selected syllabus topics before and after the course, where the delta can be considered as a learning outcome, see Fig. 1 and Fig. 2. In reviewing the learning outcomes, the course personnel is especially interested in the topics that are not that well received in order to make improvements for the next iterations.

Beforehand, MongoDB was the most unfamiliar topic to students in both campuses. A minor difference between Hervanta and Centre Campus students was detected: Hervanta folks did not know NodeJS and its middleware, whereas in Centre Campus the basics of HTTP were more intractable, indicated by high scores in cookies/sessions, authentication and security threats. Post-test questionnaire demonstrated an improvement while authentication and Mongo still remained the lowest scorers, yet the drop from 71% $\rightarrow$ 6% in poor with Mongo, and 61% $\rightarrow$ 7% with authentication shows that a lot has happened in the internalization of these topics.

4.2 The Results of the Weekly Exercises, Exam, and the Assignment

This section goes through the results students got from the weekly exercises, exams, and the assignment.

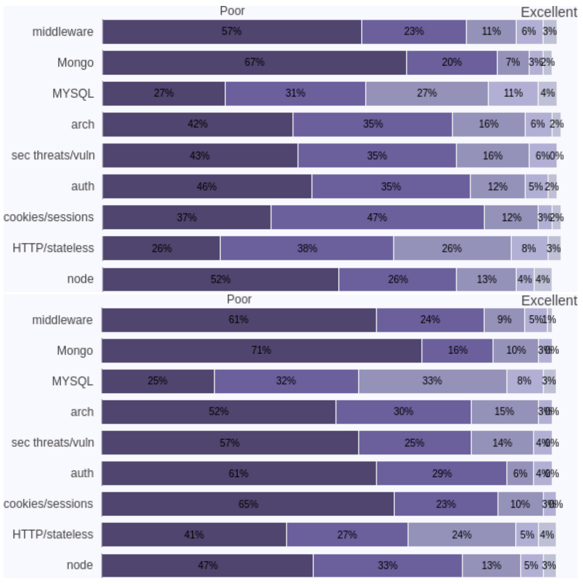

Figure 1: Prior knowledge of the course topics, Hervanta (N=191) and Centre Campus (N=88).

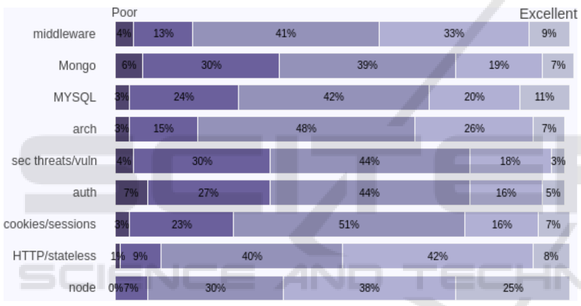

Figure 2: Learning self-estimates (post-test), all (N=144).

4.2.1 Weekly Exercises

The weekly exercises in WETO were separated into five exercise rounds, each round consisting of three exercises. An exception to this was the fifth round with five exercises. The rounds advanced from the basics of NodeJS to a comparatively complex authentication and registration implementations by modifying an existing Web application given to students. Table 1 presents the subject matters of these exercise rounds using the same categories shown in the post-test (see Fig. 2).

Table 1 shows the distribution of the points students got for each exercise round.

4.2.2 Exam

Fig. 3 illustrates the points BWA students got from the exam, 4 represents TIETA12 students' points. The online exam was the same for both student groups.

Table 1: The subject matter of five weekly exercise rounds separated to categories as shown in the post-test results.

Subject category	**1**	**2**	**3**	**4**	**5**
Middleware			X	X	X
MongoDB				X	X
MySQL				X	
Architecture					X
Security threats and vulnerabilities		X	X	X	X
Authorization and user management					X
Cookies and sessions			X		
HTTP/stateless	X				
NodeJS	X	X	X	X	X

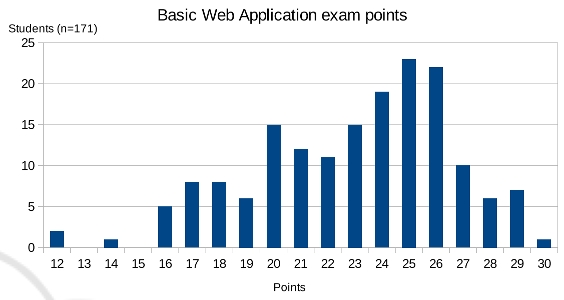

Figure 3: BWA exam points (N=171, mu=23,4, std=3,67).

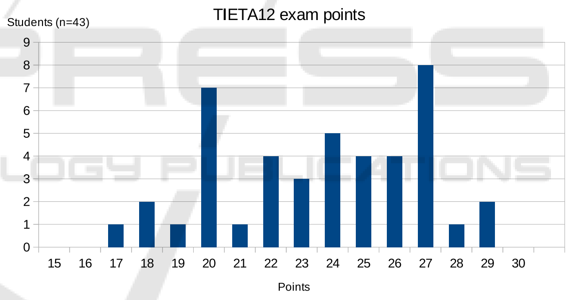

Figure 4: TIETA12 exam points (N=43, mu=24, std=3,22).

4.2.3 Assignment

Assignment grades were formed as a sum of an automated assessment and a bonus given by the course personnel. The automated assessment grade scale was [fail, 0, 1, 2, 3], and the bonus scale was [-1, 0 , +1]. The bonuses were given to a whole group, whereas the automated assessment provided separate grades for each individual. That is, the automated assessment grade varied between members in one group.

Fig. 5 shows the auto-assessed components of the assignment defined in Sec. 3.2. With the Gaussian grading, no constant limits for grades can be set beforehand, since they are relative to the performance of the whole group. The dynamic grade-level definition complicates the students' chances to check the level of their own work.

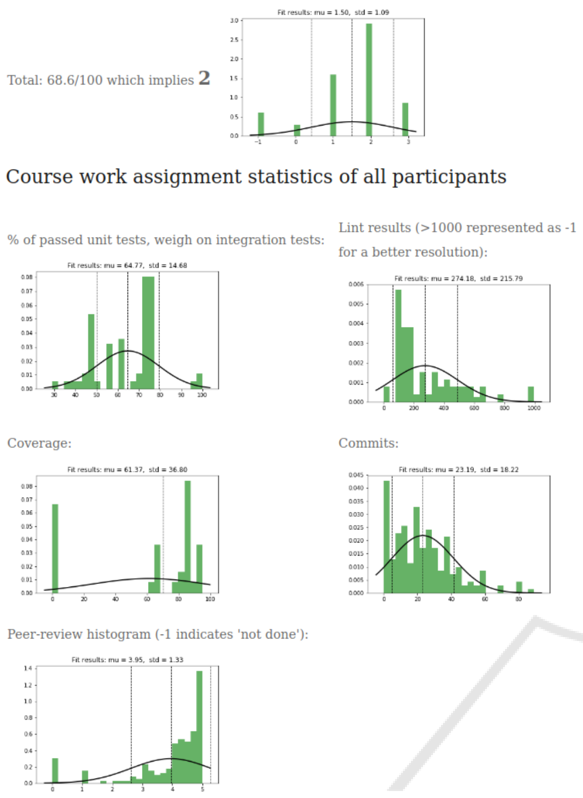

Figure 5: Example of the automatic assessment report sent to students. Each assessed component is shown in a separate graph, black curves smooth the histogram by illustrating the Gaussian distribution of all submissions.

4.3 Peer-review/Auto-assess Views

Peer-review attitudes kept steady during the research period (Hervanta, mu=2.6 std=1.0, Centre Campus mu=2.7 std=1.1); in the post-test, the same (mu=2.7 std=1.1), in Fig. 6:

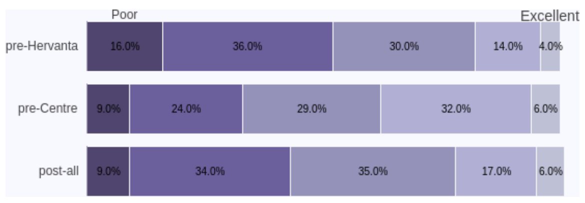

Figure 6: Pre- and post-views of peer-reviews' functionality.

WETO peer-reviews are extensively exploited in the Centre Campus; in Hervanta Campus, Plussa auto-assesses exercises with plenty of graders. Plussa's peer-review component is a relatively new feature.

In overall, the attitudes towards auto-assessing are more positive (mu=3.7, std=1.0) compared with peer-reviews, (Fig. 7), although somewhat lower rates

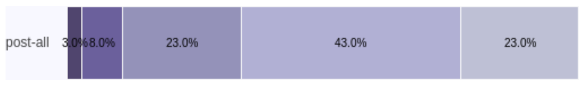

Figure 7: Views of all participants of auto-assessment (post-test only).

were anticipated due to technical problems during the auto-assessment of exercises. It can be speculated that the scores would have been better without these problems. Unfortunately, pre-test values of auto-assessment are not available.

4.3.1 Free-text Assessment Feedback

Next, the responses to the questions *'How have you benefited from peer-reviews (in any context)?'* and *'Which cons do you associate with peer-reviewing?'*.

As a summary, people value high "another eyes", that is, different perspectives, hints and correction suggestions they get from reviews. However, students with no motivation, skills or with malicious intentions lower the trust on fair assessment. The importance of feedback is well understood, a few students also point out that due to lack of time TA's chances to concentrate on each submission and give feedback can be much more constrained compared with peers. Then, peer-reviews are considered absolutely better than nothing. One's attitude about peer-reviews is determinative: it can be regarded either as a chore or a chance to learn more – including soft skills, such as communication skills.

The quality of the peer-reviews is substantially dependent on how well the grading is instructed. Good instructions picture the structure of an ideal response, which intensifies learning during the reviews. However, the review must be fit for a peer-review and not merely routine work as the next comment highlights:

> *We used peer-reviews during one math course. Since it's math, and we had very clear, definitive guidelines on reviewing, I don't think I received any significant benefits, given the clarity of the subject.*

Like math, the majority of CS exercises are closed in nature implying that the end result is known beforehand. To conclude: what can be automated ought to be automated for the sake of uniformity, consistency, and fairness. In Plussa, various graders are ready to serve. Peer-reviews are functional but for different reasons: e.g., for getting new perspectives - food for thought - learning soft skills, such as communicating, and giving constructive feedback, thus ultimately, for social cohesion. From the pedagogical point of view, peer-reviews invite students to take one step forward in Bloom's taxonomy, from applications to analyses and evaluations (Bloom et al., 1981). This idea is captured in the following quote:

Thinking of good points in a peer's solution adds tools to my own problem solving.

Students stress the quality of peer-reviews in the further development of the system.'*How would you develop or improve peer-review systems further?* ':

- Improve peers.
- Enforce penalties for poor peer-reviews and give extra credit for good peer-reviews.
- Maybe more specific instructions for feedback and giving points. "Honest attempt" really meant nothing to some reviewers.
- More clear instructions on how to give reviews and also maybe some random checks on how they are being given (and their length).
- I would penalize peer-reviews of a zero-effort, possibly even with -1 on grade. It would be harsh, but that's basically the only way to get good reviews.
- Move peer-reviews as a part of the project work. Then it would work and people would be reviewing commits before merging them into the project main branch as you do in real work. This way the importance of peer-reviews would be noticed better (and code is probably less likely to have bugs etc.).
- I think it just annoys most people because of extra work. So I personally just changed everything to be auto-assessed.

The conclusions are clear: good and conscientious reviews should be rewarded, and poor ones maybe penalized. This is important for motivation, and adds to the functionality of the entire system. In addition, the quality of the instructions is critical.

4.4 Gitlab - A Contribution Bookkeeper

Usually, group assignments are not very popular. One explanation is the uneven distribution of work: grade-sensitive students take usually the initiative and make the preponderance, while students without ambitions are tempted to freewheel. Typically, freewheeling can go unnoticed if there are no means to follow the workload and contributions within a group. By checking the number of commits, distribution of commits, and peer-review ratings, this course strove to hinder the harmful behaviour by increasing the awareness of it, at minimum. '*You get what you grade*', thus, the variation in grades within a group must reflect different workloads and peer-review ratings.

4.5 Discussion and Lessons Learned

Computer science education aims at the provision of key competencies for becoming software professionals. Compared with other university domains, CS education has a distinguishable profile. This profile sketches a picture of an ideal CS professional that is an autonomous and practical problem solver (Ylijoki, 1998). CS education has, until the very days, promised a job and not whatever job, but one that is well paid. However, to be eligible in the eyes of potential recruiters preferably requires a proven craftsmanship: e.g. published applications or other deliverables in GitHub/DockerHub. These publications function as a portfolio and their starring as a quality measure.

In university, the basic dilemma is how much it should be a vocational school versus how much theory and general study and academic skills to teach (Ylijoki, 1998). In the dilemma, enterprises and students unite in favor of immediate usefulness: employability is an important factor for both. In contrast, CS academics prefer more stable groundings based on mathematics, logic, formal methods, proofs, including a deeper look under the hood to main building blocks of programming languages.

In project-based learning that uses DevOps and Gitlab, education and assessment align more with the anticipation of enterprises and students: the industry tools and conventions prepare aptly for the future. In addition, open-ended assignments assessed by self-made tests can be considered as an application of flipped assessment, where the challenge level is fine-tuned by the students. Ready-made tests, lint and coverage reports are to be complemented with tests done by the students. Transparency of assessment increases. The authentic group assignments teach autonomy, self-regulation, and soft skills, such as negotiation skills, as teams themselves are responsible for agreeing plans and keeping up the schedule and quality. For better quality, students also suggest to deploy teams' internal code inspections, see 4.3.1.

5 CONCLUSIONS

RQ1: In flipped learning, formative and continuous feedback is a goal. To reach this with less course personnel is an oxymoron. Conjuring tricks are to increase the amount of auto-assessment and to get students to review each other's work. Automatic assessment is preferred by students since it performs consistently and fairly, and is patient. Gitlab provides a versatile API for upgrading the assessment at a new level.

In peer-reviews, well-thought and detailed enough instructions are the necessity.
RQ2: Both assessment styles have pros and cons: peer-reviews are a great asset as thought-provokers to contrast one's perspectives. At their best, peer-reviews create social cohesion, teach soft skills such as communication and negotiation skills, and elevate students to the next step of evaluation in Bloom's taxonomy. As a grader, a computer is considered more reliable than an ill-motivated, uneducated peer, but it is deficient as a soft-skill trainer.

Further Studies. Peer-reviews should comprise code inspections. Repolainen is capable of supporting this by granting report permissions. In addition, Gitlab tools function well and reliably, e.g., a wider-spread use of DevOps practices (CI, test reports, and Issue Board) are recommended in line with project-based learning. With additional resources, learning analytics with Git statistics would be a value-add. In this, there could be significant synergy in cooperating with Visual diagnosis for DevOps software development (VISDOM).

More tight-knit co-operation with the recruiting enterprises was anticipated to ensure that the university is well aware of the current industry needs, and feeds back whether the increased DevOps course supply produces better fit summer and thesis workers in the field. In addition, the self-reflective development of university LMSs using DevOps, e.g., 'infrastructure as software', would give more insight into teaching it, and improve the quality (Kohomäki, 2019).

REFERENCES

Akçayır, G. and Akçayır, M. (2018). The flipped classroom: A review of its advantages and challenges. *Computers & Education*, 126:334–345.

Arum, R. and Roksa, J. (2011). *Academically adrift: Limited learning on college campuses*. University of Chicago Press.

Bloom, B. S., Madaus, G. F., Hastings, J. T., et al. (1981). *Evaluation to improve learning*. McGraw-Hill.

Britton, E., Simper, N., Leger, A., and Stephenson, J. (2017). Assessing teamwork in undergraduate education: a measurement tool to evaluate individual teamwork skills. *Assessment & Evaluation in Higher Education*, 42(3):378–397.

Finch, D. J., Hamilton, L. K., Baldwin, R., and Zehner, M. (2013). An exploratory study of factors affecting undergraduate employability. *Education+ Training*, 55(7).

Gannod, G. C., Troy, D. A., Luczaj, J. E., and Rover, D. T. (2015). Agile way of educating. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–3. IEEE.

Graziano, K. J. (2017). Peer teaching in a flipped teacher education classroom. *TechTrends*, 61(2):121–129.

Gupta, M. (2017). Liquid workforce: The workforce of the future. *Radical Reorganization of Existing Work Structures through Digitalization*, page 1.

Haaranen, L. and Lehtinen, T. (2015). Teaching git on the side: Version control system as a course platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 87–92.

Joseph, N. (2009). Metacognition needed: Teaching middle and high school students to develop strategic learning skills. *Preventing School Failure: Alternative Education for Children and Youth*, 54(2):99–103.

Kohomäki, S. (2019). DevOps-käytännöt opetusjärjestelmien kehityksessä ja ylläpidossa. Master's thesis.

Koskinen, P., Lämsä, J., Maunuksela, J., Hämäläinen, R., and Viiri, J. (2018). Primetime learning: collaborative and technology-enhanced studying with genuine teacher presence. *International journal of STEM education*, 5(1):20.

López-Alcarria, A., Olivares-Vicente, A., and Poza-Vilches, F. (2019). A systematic review of the use of agile methodologies in education to foster sustainability competencies. *Sustainability*, 11(10):2915.

Niemelä, P. and Hyyrö, H. (2019). Migrating learning management systems towards microservice architecture.

Pintrich, P. R. (2002). The role of metacognitive knowledge in learning, teaching, and assessing. *Theory into practice*, 41(4):219–225.

Schön, D. A. (1991). *The reflective turn: Case studies in and on educational practice*. Teachers College Press New York.

Sennett, R. (2011). *The corrosion of character: The personal consequences of work in the new capitalism*. WW Norton & Company.

Sointu, E., Valtonen, T., Kankaanpää, J., Hyypiä, M., Heikkinen, L., and Hirsto, L. (2019). Ingredients for a positive view of flipped classroom in higher education. In *EdMedia+ Innovate Learning*, pages 1672–1679. Association for the Advancement of Computing in Education (AACE).

Tan, C., Yue, W.-G., and Fu, Y. (2017). Effectiveness of flipped classrooms in nursing education: Systematic review and meta-analysis. *Chinese Nursing Research*, 4(4):192–200.

Tandogan, R. O. and Orhan, A. (2007). The effects of problem-based active learning in science education on students' academic achievement, attitude and concept learning. *Online Submission*, 3(1):71–81.

Toivola, M. (2020). Flipped assessment: A leap towards assessment for learning.

Toivola, M., Peura, P., and Humaloja, M. (2017). Flipped learning in Finland.

Vogler, J. S., Thompson, P., Davis, D. W., Mayfield, B. E., Finley, P. M., and Yasseri, D. (2018). The hard work of soft skills: augmenting the project-based learning experience with interdisciplinary teamwork. *Instructional Science*, 46(3):457–488.

Ylijoki, O.-H. (1998). Akateemiset heimokulttuurit ja yliopistoyhteisön itseymmärrys. *Tiedepolitiikka: Edistyksellinen tiedeliitto ry: n julkaisu 23 (1998): 3.*