Experience Report on a Student-Organized AI Course

Sebastian Krings
sebastian.krings@uni-duesseldorf.de
Heinrich Heine University
Düsseldorf, Germany

ABSTRACT

As curricula are mostly centered around the specializations of faculty members, students are often seen purely as consumers of knowledge.

In contrast, in the course presented in this experience report, we gave as much control to the attendees as possible. As the course was overbooked, we had to use an innovative format to search strength in numbers rather than limiting the number of participants. We crowdsourced lectures, practices and even the exam. This allowed our students to become (co-)producers rather than consumers of knowledge.

Along with presenting our methodology, we evaluate whether our approach was successful and whether it could serve as an example for others.

CCS CONCEPTS

• **Social and professional topics → Computing education programs**; *Model curricula*; *Information systems education*; *Student assessment*; *Adult education.*

KEYWORDS

student self-organization, artificial intelligence, experience report, course design, crowdsourcing

ACM Reference Format:
Sebastian Krings. 2022. Experience Report on a Student-Organized AI Course. In *Proceedings of the 27th ACM Conf. on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524805

1 INTRODUCTION

The *Software Engineering and Programming Languages* group at the University of Düesseldorf teaches courses on logic programming, in particular using Prolog [3] as a programming language. While the group's research is centered around software verification techniques, Prolog is more commonly used in artificial intelligence, mostly to develop expert systems and for language processing. Additionally, Prolog can be used to implement common AI algorithms for games quite easily as we show in various examples throughout our lectures.

In consequence, we had several requests by students to extend our lecture portfolio by a course on artificial intelligence. However, at that time, we did not have the teaching resources to create a full-blown lecture and thus decided on a lightweight seminar to fulfil our student's request.

In particular, we intended to broaden the scope beyond classical applications of Prolog in AI and take into account modern machine learning techniques, computer vision as well as ethical questions regarding artificial intelligence. All the new topics would have to be prepared and presented by students, leading to discussions where appropriate. In particular, we wanted to increase student involvement as much as possible, since it is linked with learning success [2, 6]. The seminar was available as an elective course in the CS curriculum. The workload was indicated as 5 or 7.5 ECTS credit points depending on exam regulations for individual students.

However, we failed to set up a limit on the number of participants, leading the course being overrun by slightly more than a hundred registered students. Obviously, this is way too much for a classical seminar, leaving us with two options:

- reducing the number of participants, ultimately disappointing numerous students, or
- switch to a new course format that embraces the high number of participants by crowdsourcing workload and, hopefully, creating interesting results.

As a result, we had to replan the course, getting rid of the classical seminar format and switch to a student-organized format in the sense of a "Contributing Student Pedagogy" [7]. The new format and how it was realized in practice will be described below. As we deemed the course a success, the format was reused a year later. Below, we will report on the general format and how the two executions of the course were evaluated.

The rest of the paper is structured as follows: Section 2 gives an overview over the course and its context, e.g., participants. Following, we discuss supporting applications and E-learning techniques in Section 3. A major point of criticism against student-organized content, namely how to assure its quality, is addressed in Section 4. Evaluation results based on student feedback is discussed in Section 5. We conclude with future work and overall conclusions.

2 COURSE OVERVIEW

2.1 Participants

Below we give an overview over the participants of the course as far as the course evaluation permits. Even though we had more than a hundred students attending each of the two course iterations, only a very small number of students answered the questionnaire: 19 in the first iteration and 11 in the second one. Hence, the data presented here and in Section 5 only represents the overall attendees to some extent.

The course was attended by a diverse group of students, across different semesters. While the course is aimed at the Bachelor's

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9201-3/22/07.
https://doi.org/10.1145/3502718.3524805

Table 1: Bachelor / Master Students

	first iteration	second iteration
bachelor	78.9%	100%
master	26.3%	27.3%

Table 2: Current Semester of Attending Students

semester	first iteration	second iteration
4	26.3%	45.5%
5	5.3%	18.2%
6	36.8%	9.1%
7	0%	18.2%
8	10.5%	9.1%
9	5.3%	0%
10	10.5%	0%

Table 3: Female / Male Students

	first iteration	second iteration
female	23.1%	11.1%
male	76.9%	88.9%

level, we allowed both Bachelor and Master students to attend. By doing so, we reacted to requests of a number of Master students who finished their Bachelor's degree before the course existed. As a result, roughly a fourth of attending students were pursuing some kind of Master's degree as depicted in Table 1. Keep in mind, that students might pursue different degrees simultaneously, e.g., a Bachelor and a (different) Master. As a consequence, the values do not add up to 100% as one might expect.

In consequence, students' semesters of study range from four to ten, as depicted in Table 2. Most students are inside their standard period of study, attending the course between the fourth and sixth semester. Again, students might be attending different degree programs at the same time, thus being counted for different semesters of study as well. As above, the values might thus not add up to 100% as expected.

Sadly, the standard student evaluation at the University of Düsseldorf does not provide us with correlations between degree program and semester of study. Thus, we cannot draw further conclusions.

As shown in Table 3, in the first iteration, roughly a quarter of the students was female. This is in line with the general number of female students of computer science at that time. In the second course execution, the percentage of female students dropped by roughly 10%. So far, we have no explanation for this change.

Summarizing, we had a very diverse group of participants, including all levels of background knowledge and prior study experience. This suggested, that we would also have to deal with different level of engagement and participation [13].

2.2 Course Outline

To handle both diversity and the large number of students present, we decided to use the first lecture to discuss possible improvements to the standard seminar format and to find areas of interest suitable for most attendants.

In particular, we tried to foster the idea, that we are all in it together and have to work collaboratively in order to enjoy a successful course despite the challenges. This is in line with the idea of collaborative learning [17], in which groups consisting of students of different performance levels work towards a common goal. Collaborative learning is associated with a number of possible benefits, ranging from social and psychological to academic advantages [18].

Furthermore, considering students as "co-producers" rather than consumers of education and knowledge has been suggested and linked to increased student satisfaction and outcome [12, 16].

With our initial goal being a successful course, we tried to identify the sub-goals that will lead us to it, all in collaboration with our students. Together, we came up with the following set of tasks needed to be done for a successful course:

- select initial coverage of course topics,
- creation of a script,
- lecture design, i.e., the actual teaching,
- creating tasks and exercises for repetition and exam preparation, and
- selecting tasks for the exam, marking and grading.

While the last item had to be done by the lecturers for obvious reasons, all other items were given into student's responsibility. We predefined a selection of possible topics to be covered throughout the seminar:

- informed and uninformed search algorithms,
- search algorithms for games,
- expert systems,
- constraint solving,
- inductive logic programming,
- object recognition,
- speech recognition,
- machine learning,
- evolutionary algorithms,
- decision trees,
- concept of intelligence and perception,
- ethical concerns and limits of artificial intelligence.

The predefined selection of topics ensures that even though most of the course was student-controlled, all important topics would be covered. In particular, the selection is in line with typical AI course contents [26].

For each topic, students could decide between preparing a chapter for the script (i.e., write a seminar paper) and preparing a lecture (i.e., give seminar presentation). In addition, both groups should create tasks for repetition and exam preparation and review other contributions in a peer-review. Of course, during later iterations of the course, students had to improve the existing script and papers rather than creating new ones.

In particular, we wanted to make sure that all tasks available foster a shift from self-explanation to interactive explanation as defined by Ploetzner et al. [22]. As the original article does not suggest that one is more efficient than the other, ensuring all students have both perspectives on explanation seemed beneficial to us.

While the preparatory tasks foster self-explanation, presentation and review support interactive explanation. As students could focus more on delivering a live lecture or on delivering script content and tasks, interactive explanation was included both for those selected for presentation and for those students staying in the background.

This approach allows for a relatively broad selection of tasks for each student to chose from. Having a multitude of options for participation, ranging from ones with more publicity (e.g., give a lecture or presentation) to ones that can be done at home, helped us to offer options suitable for different kinds of students. Thus, it helped us to reduce the effects of the consolidation of responsibility [15], where certain students become the center of attention, dominating the course and its interaction.

However, with increased diversity in the tasks, equity becomes more questionable. We tried to keep both workload and learning outcome roughly identical throughout different tasks:

- We could, for instance, increase or decrease the number of assigned reviews, depending on the scope of the assigned topic and whether a lecture was to be prepared or a seminar paper had to be written.
- We advised excluding certain aspects of a topic or to go more in-depth depending on the work done so far.
- Learning outcomes should be harmonized by reviewing lectures and scripts (i.e., reviewing the work created by others) or by answering repetition question created by others.

Yet, individual workload is both hard to measure and to control. As a consequence, some students had a considerable higher workload than others (cf. Section 5).

Sadly, we have no usable data on how students evaluated fairness of assignments, both w.r.t. kind of work and assigned / selected topic. From our observation, there was no discontent regarding equity and there was none voiced during evaluations. However, this is anecdotal at best and needs to be evaluated further in coming iterations of the course.

All tasks were supported using different E-Learning techniques and other online applications, which will be discussed in detail in Section 3.

The course ended with a classical exam used for summative evaluation, mostly based on modified and extended versions of the tasks submitted by the students. Additionally, we performed a formative evaluation of course involvement, taking into account preparation and presentation task as well as participation in the online parts. The way of grading was of course again suggested by and discussed with the students.

To our discontent, the overall outline was not fully in accordance with the learning outcomes given in the module handbook. The soft skill required for discussion and collaborative preparation were not requested there and were not fully taken into account for grading, effectively causing issues with the course's constructive alignment [1].

However, the additional skills acquired and trained are highly requested by industry. Especially working in and managing projects as well as working collaboratively are often listed as experiences computer science students are lacking [23, 24].

3 E-LEARNING AND SUPPORTING APPLICATIONS

To support our course we used ILIAS [14], a learning management system popular in Germany. ILIAS was mainly used for coordination and self-evaluation. Furthermore, we relied on git, GitHub and LaTeX for cooperative script creation.

3.1 ILIAS Message Boards

ILIAS features message boards that can be added to course home pages. We decided to use three message boards: one for organizational question, one for questions regarding course topics and one to share and discuss further related topics and literature. The first two were used quite seldom, mostly for questions regarding the exam. The third one was filled by us quite frequently and was read by most of the students. However, ILIAS reports every link as read as soon as it is opened by a student, i.e., we have no way to tell whether the additional material was thoroughly read. Given that it was only seldom discussed we assume it was not. In summary, the message boards did not add much to the course, especially given the already high level of participatory elements.

3.2 Self-Evaluation using ILIAS

A substantial collection of tasks has been created in ILIAS, most of them suitable for automatic correction. We were quite content with the options offered by ILIAS (e.g., fill-in-the-blank texts, matching and ordering tasks, numeric questions, image map questions) and the options for automatic grading and feedback.

However, it was very complicated to set up access rights in a way that would allow students to directly create course content on ILIAS. While we could assign rights to individual tasks, working on different tasks on and off in groups could not efficiently be allowed without permitting basically anything. In consequence, we had to manually create ILIAS content from students' input, causing a high workload for the lecturers, as they had to manually create quizzes and other course content from the students' input. Overall, self-assessment was well received by the students, most tests were run shortly before the following lecture and, of course, in the week before the exam took place.

3.3 Selecting Papers for Script Inclusion

Due to the number of participants, we often had different students write seminar papers on the same topic. In consequence, we had to come up with a way to select the ones to be included in the script.

We did so using ILIAS' capabilities for peer review, automatically assigning all papers submitted to a selection of student reviewers. To ensure common criteria for the reviews, we discussed how and why peer review is performed at scientific conferences and what it is supposed to ensure. In particular, we made clear that the script should serve as a learning aid for the exam and that there will still be room for improvement afterwards, i.e., reviews should help the author improve. The reviews were mostly submitted on time and were helpful (cf. Section 4).

3.4 Script Creation

The different seminar papers written by students have been incorporated into a single script. To write the script collaboratively, we used git and GitHub. While these tools were initially created for distributed software development, adopting certain tools and workflows for the course was highly beneficial:

First, most students were already familiar with git and GitHub. Furthermore, LaTeX will be used by most students for their thesis.

Second, the access model of GitHub and the common git workflow suits well for our tasks. Students could create their individual copy of the script and work on it, i.e., extend sections, add and modify examples, etc. Once satisfied with the results, students could submit a request for integration into the main repository (called a pull request). This request is not executed automatically. Rather, a request has to be approved by two other students, which can check for typos or other errors. As soon as a request has been approved, lecturers could integrate the changes into the script with the click of a button. Note that a pull request can be declined, ideally including the reasons for the decision. In that case it is reported back to the initial writer who can then improve the changes made to the script and resubmit the request for integration. Again, the process enforces the ideas of collaborative learning by enforcing common decisions rather than individual ones [17].

Third, the script can be built and uploaded using GitHub's continuous integration system. This ensures that the latest version of the script is readily available for download and ensures updates easily propagate to the students.

Overall, the resulting script is roughly 150 pages long and includes documentation and examples to most of the course topics. It is completely written by our students, with only minor editorial work by the lecturers. Later iterations of the course will of course use the script and thus do not have to write extensive seminar papers but rather extend, unify and improve existing chapters.

3.5 Reliance on CS Background

For this to work efficiently, being able to rely on knowledge taught in other CS courses was highly beneficial. Collaboration using Git and GitHub is taught in a programming practice course in the second semester. At the same time, concepts such as branching, merging and code reviews are known to most of the students and usually no further explanation was needed. Furthermore, LaTeXwill be used for other reports and theses as well.

While our approach was easy to realize in a CS course, the same approach could be transferred to non-CS courses with a little effort:

- Git and GitHub would have to be replaced by a toolchain more suitable for non-developers, e.g., Google Docs, Collabora or a plain wiki system.
- Peer reviews could be performed using common platforms such as ILIAS or Moodle. Simultaneously, peer review could be used to replace the review workflow based on pull requests.
- There are several other software systems to support student-generated assessments, e.g., Peerwise [4]. These systems typically can be used without a CS background.

4 ASSURING LEARNING OUTCOMES

One of the most obvious concerns with a student-organized course format is quality assurance. Intended learning outcomes should be reached and students should reach a certain depth rather than only staying on the surface of the topics we listed in Section 2.2.

Obviously, the format is limited by the ability of the students, who usually have only limited knowledge about the subjects to present as well as about pedagogic practices. As a result, quality of teaching might be reduced despite the fact that we introduced peer review for initial quality assurance.

Regarding the quality of peer assessments, a study performed on classes on introductory programming confirms that peer assessments can be as effective as tutor feedback if done right [9]. In the context of software engineering, Hamer et al., studied the differences between peer and tutor feedback [8]. They report that while tutor feedback is of higher quality overall (e.g., due to being more precise), differences are negligible in areas such as offering advice.

Pirttinen et al. [21] recently revalidated the previous findings of Hamer et el. [9]. They used a tool called CrowdSorcerer [20] to compare peer reviews of crowdsourced programming assignments. One of their major results is that novices can review as well as more experienced programmers. The same holds for creating assignments. Even though general programming task do not fully compare to the task crowdsourced in our course, this provides further evidence that our approach is sensible.

While we cannot claim to be representative, we were happy with the quality of the reviews performed by our students. Most reviews where thorough, discussed both technical and stylistic issues and made suggestions for improvement. While not all reviews reached the desired quality, assigning multiple reviews per submission usually ensured receiving a high-quality review.

Student-generated questionnaires have been shown to be effective learning tools in the context of molecular biology [10] as well as physics, chemistry and biology [11]. Furthermore, creating questionnaires is improving students understanding as well [5]. Thus, both the content creators and the other students benefit.

To our knowledge, no representative study on the effectiveness of student-generated content in the context of AI courses has been performed yet.

In summary, the lecturer's knowledge remains the foundation of a successful course [25]. The overall question is to what extent we have to use it for direct teaching rather than supporting tasks.

5 STUDENT EVALUATION

The course was evaluated using questionnaires, given out in the last few weeks. Invitations to rate the course were sent out by mail to all students enrolled.

The questionnaire used for the default course evaluation is standardized and could not be modified by us. Thus, we had no influence on the questions asked or on the scale used. We could of course have provided our own additional questionnaire, but wanted to avoid bothering our students with two evaluations for the same course.

As stated above, only a limited number of students participated in the evaluation: 19 for the first iteration and 11 for the second

Table 4: Course Evaluation, mean ($\overline{a}$) and median ($\tilde{m}$), scale from 1 (= total agreement) to 5 (= total disagreement)

	first iteration		second iteration	
	$\overline{a}$	$\tilde{m}$	$\overline{a}$	$\tilde{m}$
course is well-structured	2.3	2	1.5	2
course material is helpful	2.6	2	1.8	2
lecturer explains well	2.2	2	1.6	1
lecturer addresses questions	1.5	1	1.2	1
lecturer is motivated	1.6	1	1.2	1
satisfied with the course	2.5	2	2	2
was interested in topic before	1.3	1	1.3	2
learning outcome was high	3	3	2.1	2
lecturer support is helpful	1.6	1	1.5	1
course gives a good overview	2.2	2	1.6	2
good mixture of knowledge transfer and discussion	2.5	2	1.7	2

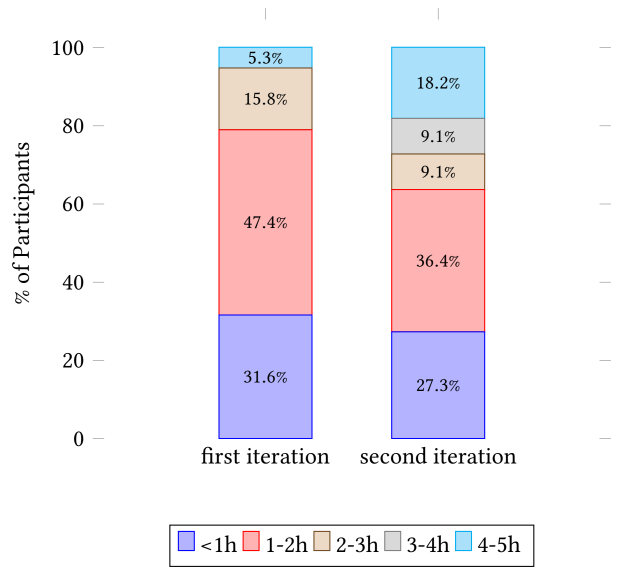

Figure 1: Weekly Time Spend for Preparation and Post-processing

one. Evaluation results were presented to and discussed with the students.

Table 4 shows the answers to the evaluation questions, giving both mean and median on a scale from 1 (= total agreement) to 5 (= total disagreement).

Overall, the course was well received, especially considering the later executions. Course satisfaction ranges from 2 to 2.5 without much delta between different iterations.

While the course was regarded as somewhat unstructured in the first iteration, the existing script made structuring both easier and more obvious in the second one. Of course, the existing script increases the helpfulness of course material as well, as the second question suggests. As interest and coverage of topics shifted, the script was able to cover more detail than individual lectures. In consequence, later course iterations tend to give a broader overview of AI.

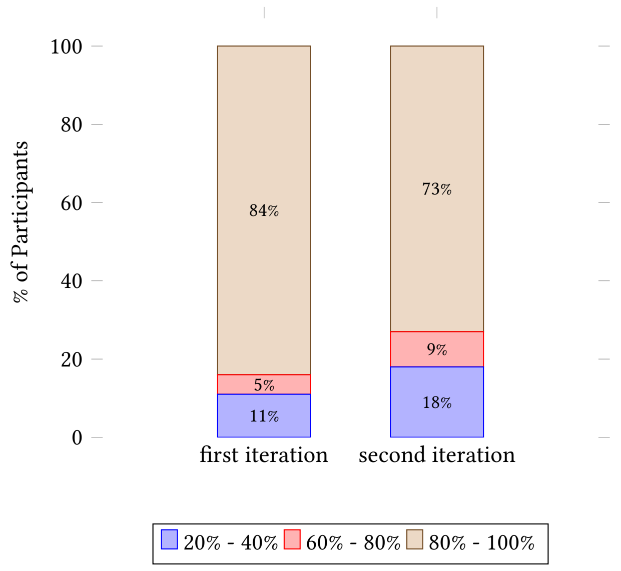

Figure 2: Percentage of Lectures Attended

Interestingly, the overall learning outcome is considered higher for those course iterations that only had to improve the script rather than write it from scratch. For now, we can only suspect that writing consumed a lot of time not available for other topics anymore.

However, as show in Fig. 1, students participating in the first course iteration did not spend significantly more time on preparation (i.e., preparing talks or seminar papers) and post-processing (i.e., repeating content, doing exercises and learning for the exam). While in the first course iteration numerous students (47.7%) only spend one to two hours a week, in the second one only about 37% of students did so. The number of students spending two to three hours a week dropped from 15.8% to 9.1% in the second one. Overall, time spend for the course increases throughout the years, even though script writing was mostly done.

Higher course attendance cannot be the reason for the increased time requirements as shown in Fig. 2. Rather, attendance dropped slightly during the years.

6 FUTURE WORK

For the coming iterations of our AI course, we want to focus on different aspects. First, the argument against giving all control to the students was that we would not be able to ensure an appropriate learning outcome. This is somewhat in alignment with the empirical evaluation we performed in Section 5, where students rated the learning outcome as 3.0 and 2.1 (with one being the best, five the worst score). While ratings increased in recently, we need a more thorough and more objective evaluation, relying on pre- and post-testing of desired outcomes.

Additionally, course evaluation suggested that we should aim for a higher amount of discussion, rather than knowledge transfer. To do so, we want to take some tasks collected for ILIAS and use them as a starting point for peer instruction questions, following the method outlined by Eric Mazur [19].

Furthermore, as discussed in Section 5, we do not fully understand how the different tasks contribute to the number of lectures attended and to the time students had to spend in preparation and post-processing. In the future, we would like to evaluate in more detail to what extent writing vs. improving a script contributes to learning outcomes and time requirements.

Finally, we need to think of ways to continue with the course, taking into account that material is more and more complete and mature. In consequence, the very interactive and participatory course tends to degenerate into a classical lecture.

7 CONCLUSION

Summarizing, we have presented a student-run course on artificial intelligence. During planing and execution, we realized different aspects, that we assume transfer to other courses:

- Crowdsourcing can be quite powerful. We managed to include a diverse range of topics appropriately and cover them in the script and tasks. However, course evaluations suggested that learning outcome could be improved. Thus, we have to evaluate to what extent shifting control hinders learning, e.g., because students spend too much time on course management.
- Cooperative writing and script creation is easy using online platforms. While GitHub is a natural choice in CS, we could easily have used other online editors.
- Being in control appears to be highly motivating for students, as it enables them to transition away from being knowledge consumers only. While we do not have proper data to compare crowdsourced to regular courses on that matter, the high motivation despite the high workload (cf. Section 5) points in that direction.

In conclusion, while we initially had doubts against a mostly student-organized course, we are very content with the successful outcome. An impressive collection of tasks and lectures has been created by the participants. At the same time, students wrote chapters to a lecture script and composed them based on peer-review, resulting in a small script on AI that can be improved continuously throughout upcoming iterations.

The benefits we observed are in line with many other case studies on crowdsourcing in computer science. For an initial overview of those, see Hamer et al. [7].

We believe that the overall approach extends beyond teaching AI and even beyond teaching CS. While it can be implemented quite easily by relying on tools and workflows known to CS students, proper replacements are available. In fact, tools such as Peerwise [4] do not rely on any CS background and can be used throughout disciplines.

We suspect that using tools, workflows and approaches common in a field to realize a student-led course format increase the chance of success, as students can focus on the other aspects of a yet unknown format.

ACKNOWLEDGMENTS

First, we would like to thank all participating students for being curious and willing to experiment. The same holds for our supervising professor, who allowed us to play around rather than sticking to what we always did. In addition, we thank Philipp Körner for many fruitful discussions regarding our approaches to teaching both related to the course presented here and to CS in general. Additionally, we would like to thank the didactics department of the Heinrich Heine University for their support and advise.

REFERENCES

[1] John Biggs. 1996. Enhancing Teaching through Constructive Alignment. *Higher Education* 32, 3 (1996), 347–364.

[2] John Bransford. 1979. *Human cognition: Learning, understanding, and remembering*. Thomson Brooks/Cole.

[3] Alain Colmerauer and Philippe Roussel. 1996. History of Programming languages—II. ACM, New York, NY, USA, Chapter The Birth of Prolog, 331–367.

[4] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise. In *Proceedings of the 8th International Conference on Computing Education Research (Koli '08)*. Association for Computing Machinery, New York, NY, USA, 109–112.

[5] Paul Denny, Ewan Tempero, Dawn Garbett, and Andrew Petersen. 2017. Examining a Student-Generated Question Activity Using Random Topic Assignment. In *Proceedings ITiCSE (ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 146–151.

[6] Linda Marie Fritschner. 2000. Inside the undergraduate college classroom: Faculty and students differ on the meaning of student participation. *The journal of higher education* 71, 3 (2000), 342–362.

[7] John Hamer, Quintin Cutts, Jana Jackova, Andrew Luxton-Reilly, Robert McCartney, Helen Purchase, Charles Riedesel, Mara Saeli, Kate Sanders, and Judithe Sheard. 2008. Contributing Student Pedagogy. *SIGCSE Bull.* 40, 4 (nov 2008), 194–212.

[8] John Hamer, Helen Purchase, Andrew Luxton-Reilly, and Paul Denny. 2015. A comparison of peer and tutor feedback. *Assessment & Evaluation in Higher Education* 40, 1 (2015), 151–164.

[9] John Hamer, Helen C. Purchase, Paul Denny, and Andrew Luxton-Reilly. 2009. Quality of Peer Assessment in CS1. In *Proceedings of the Fifth International Workshop on Computing Education Research (ICER '09)*. Association for Computing Machinery, New York, NY, USA, 27–36.

[10] Dale Hancock, Nicole Hare, Paul Denny, and Gareth Denyer. 2018. Improving large class performance and engagement through student-generated question banks. *Biochemistry and Molecular Biology Education* 46, 4 (2018), 306–317.

[11] Judy Hardy, S. Bates, M. M. Casey, Kyle W. Galloway, Ross K. Galloway, Alison E. Kay, P. Kirsop, and H. McQueen. 2014. Student-Generated Content: Enhancing learning through sharing multiple-choice questions. *International Journal of Science Education* 36 (2014), 2180–2194.

[12] Thorsten Hennig-Thurau, Markus F. Langer, and Ursula Hansen. 2001. Modeling and Managing Student Loyalty: An Approach Based on the Concept of Relationship Quality. *Journal of Service Research* 3, 4 (2001), 331–344.

[13] Jay R Howard, George H James III, and David R Taylor. 2002. The consolidation of responsibility in the mixed-age college classroom. *Teaching Sociology* (2002), 214–234.

[14] ILIAS open source e-Learning Society. 2022. ILIAS. http://www.ilias.de, Last accessed on 2022-03-21.

[15] David Karp and William C. Yoels. 1976. The college classroom: Some observations on the meanings of student participation. *Sociology & Social Research* 60 (07 1976), 421–439.

[16] Theuns Kotzé and P J. du Plessis. 2003. Students as 'co-producers' of education: A proposed model of student socialisation and participation at tertiary institutions. *Quality Assurance in Education* 11 (12 2003), 186–201.

[17] Marjan Laal and Mozhgan Laal. 2012. Collaborative learning: what is it? *Procedia - Social and Behavioral Sciences* 31 (2012), 491–495. World Conference on Learning, Teaching & Administration - 2011.

[18] Marjan Laal, Azadeh Sadat Naseri, Mozhgan Laal, and Zhina Khattami-Kermanshahi. 2013. What do we Achieve from Learning in Collaboration? *Procedia - Social and Behavioral Sciences* 93 (2013), 1427–1432.

[19] Eric Mazur. 1996. *Peer Instruction*. Prentice Hall.

[20] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing Programming Assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. Association for Computing Machinery, New York, NY, USA, 326–331.

[21] Nea Pirttinen, Vilma Kangas, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Analysis of Students' Peer Reviews to Crowdsourced Programming Assignments. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. Association for Computing Machinery, New York, NY, USA.

[22] Rolf Ploetzner, Pierre Dillenbourg, Michael Preier, and David Traum. 1999. Learning by explaining to oneself and to others. *Collaborative learning: Cognitive and computational approaches* 1 (1999), 103–121.

[23] Alex Radermacher and Gursimran Walia. 2013. Gaps Between Industry Expectations and the Abilities of Graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 525–530.

[24] Alex Radermacher, Gursimran Walia, and Dean Knudson. 2014. Investigating the Skill Gap Between Graduating Students and Industry Expectations. In *Proceedings of the 36th International Conference on Software Engineering (ICSE Companion 2014)*. ACM, New York, NY, USA, 291–300.

[25] Lee Shulman. 1987. Knowledge and Teaching: Foundations of the New Reform. *Harvard Educational Review* 57, 1 (1987), 1–23.

[26] Michael Wollowski, Robert Selkowitz, Laura E. Brown, Ashok Goel, George Luger, Jim Marshall, Andrew Neel, Todd Neller, and Peter Norvig. 2016. A Survey of Current Practice and Teaching of AI. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 4119–4124.