

Question Generation Workflow: Incorporating Student-generated content and Peer Evaluation

by

Chungmin Lee

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

© Massachusetts Institute of Technology 2020. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
May 12, 2020

Certified by
Rob Miller
Distinguished Professor
Thesis Supervisor

Accepted by
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Question Generation Workflow: Incorporating Student-generated content and Peer Evaluation

by

Chungmin Lee

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Most classroom settings offer limited opportunities for students to engage with the class material. Furthermore, students are not able to receive timely feedback on their work due to limited number of staff who are able to provide comments. With increased student work to grade, the time it takes for staff to grade increases. In this thesis, we propose a question-generation workflow that addresses these issues. In this workflow, students have the opportunity to learn class material more in depth by creating their own multiple-choice questions (MCQs) on class material. As part of the workflow, students also answer and leave feedback on MCQs written by their peers. Both the student-written question and the feedback on those questions go to staff for review and staff can rely on the feedback to determine more quickly the quality of a student-written question. The application of this workflow with Questionable, a web application that can handle both student and staff needs as part of this process, in a computer science course at MIT has shown promising results of students engaging with the course material and improved grading experience for staff.

Thesis Supervisor: Rob Miller

Title: Distinguished Professor

Acknowledgments

I would like to first thank my thesis advisor Rob Miller for all of his help and support throughout the last year. Writing this thesis would not have been possible without your patience, generosity and advice. Your willingness to always provide help when needed has been inspiring and encouraging. I have truly learned a lot working with you and will carry all that I have learned in the places I will be.

The biggest anchor during my career at MIT has been and will always be my family. I would like to thank my parents and my siblings, Christina and James, for their unconditional support and love, and for believing in me more than I believe in myself. I am always so grateful for everyone's encouragement. Your kind words have helped me push through tough times, and I am forever thankful for that.

Thank you to my dearest friends for the constant support and care. I will not forget the help you all have given me throughout the semesters, especially during these recent times. Thank you for reminding me to push my limits and realize my potential. A special thank you to Mary Z. Zhong for being my grad school buddy, for always supporting me and most importantly, for getting coffee with me almost every morning from Area Four. I will always cherish our friendship and memories at MIT.

I would like to finally thank the 6.031 staff and the members of Usable Programming, all of whom are also part of staff, for helping me out throughout various stages of my research, from the design to the naming of my system, Questionable. It has been fun to share the excitement of teaching with everyone. I will definitely miss having our UP standups and all the laughs we shared at meetings.

Contents

1	Introduction	13
2	Related Work	17
2.1	Learning through Testing Effect	17
2.2	Student generated content platforms: Codewrite, Peerwise, StudySieve	18
2.2.1	Codewrite: Sharing coding exercise	18
2.2.2	Peerwise: Sharing Multiple-Choice Questions (MCQs)	19
2.2.3	StudySieve: Free-response questions	20
2.3	Item Response Theory (IRT)	22
2.3.1	Assumptions in IRT	22
2.3.2	Item Response Function (IRF)	23
3	Design & Implementation	27
3.1	Proposed New Question Workflow	27
3.1.1	Student-side changes	28
3.1.2	Staff-side changes	29
3.1.3	Application of Question Workflow in 6.031	29
3.2	Questionable: web application for question workflow	30
3.2.1	Overall Structure of Questionable	30
3.2.2	Design of Student-side UI	31
3.2.3	Student-side Implementation	37
3.2.4	Staff-side UI	38
3.2.5	Staff-side Implementation	43

4	Evaluation	45
4.1	Application of IRT on 6.031 Data	46
4.1.1	Processing Student Response Data	47
4.2	IRT Results on 6.031 Nanoquizzes	48
4.2.1	Ability (θ) Estimation Results	48
4.2.2	Item Parameters (a and b) Estimation Results	50
4.3	Student-generated Questions	52
4.3.1	Categorization of Questions	52
4.3.2	IRT Estimation of Student-written Question Parameters . . .	54
4.3.3	Qualitative observations	59
4.4	Level of student engagement	60
4.5	Changes in Grading Experience for Staff	60
5	Discussion	63
5.1	Conclusion	63
5.2	Future Work	66

List of Figures

2-1	Item Characteristic Curve (https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory)	24
2-2	ICCs with different discriminability. Item 1 has a steeper slope (higher a value) than Item 2. (https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory)	25
2-3	ICCs with different difficulty values of -1, 0 and 1, respectively, from left to right. (https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory)	26
3-1	Page 1 of the makeup form in Questionable for Students	32
3-2	Page 2 of the makeup form in Questionable for Students	33
3-3	Answers and explanation after not answering question incorrectly	35
3-4	Feedback generated when student answers question correctly	35
3-5	Question submission page of student-side of Questionable	36
3-6	List of all makeups on staff-side of Questionable	40
3-7	List of all makeups on staff-side of Questionable	40
4-1	Plot of Percentage of Nanoquiz Questions Students Correctly Answered and Their Theta Values Estimated from Nanoquiz Responses (6.031 Fall Semester 2019)	49
4-2	Plot of Final Grade for 6.031 and Their Theta Values Estimated from Nanoquiz Responses	49
4-3	Plot of Estimated Item Difficulty (b) and Estimated Item Discriminability (a) of Nanoquiz Questions from 6.031 Fall Semester 2019	51

4-4 Plot of Estimated Item Difficulty (b) and Estimated Item Discriminability (a) of Student-generated Questions from 6.031 Fall Semester 2019	54
--	----

List of Tables

4.1	Minimum, Maximum and Median of the Estimated Item Parameters of the Nanoquiz Questions.	52
4.2	Minimum, Maximum and Median of the Estimated Item Parameters of the Student-generated Questions.	54
4.3	Question Categories Percentage By a ranges of Student-generated Questions	56
4.4	Question Categories Percentage By b ranges of Student-generated Questions	56
4.5	Question Categories Count By a ranges of Student-generated Questions	56
4.6	Question Categories Count By b ranges of Student-generated Questions	57
4.7	Question Categories Percentage By High a , High b and Low a , Low b of Student-generated Questions	57
4.8	Student-generated Question Count by Question Characteristics (Ill-formatted, low grades, simple code execution) Across a Values	58
4.9	Student-generated Question Count by Question Characteristics (Ill-formatted, low grades, simple code execution) Across b Values	58
4.10	Student-generated Question Distribution by Question Characteristics (Ill-formatted, low grades, simple code execution) Across a Values . .	58
4.11	Student-generated Question Distribution by Question Characteristics (Ill-formatted, low grades, simple code execution) Across b Values . .	59
4.12	Grade Breakdown by Cause for Borderline and Minus Grades	59
4.13	Playtester Attempts Statistics	60

4.14 Playtester Correct Answer Rate Statistics (measures the rate a student ends up answering a question correctly.) 60

Chapter 1

Introduction

The usual work flow in a traditional classroom involves mainly passive work from students and does not offer many opportunities for students to actively engage with the class material. In most class settings, student involvement with the class material is limited to doing the given homework and studying for exams to demonstrate their knowledge. Any kind of studying beyond the given class framework is often up to the students to design and execute. As a result, students may continue to study by just passively rereading notes and textbook. They face reduced incentives to seek other ways to actively learn and engage with the class material and the learning process. Examples of active learning include discussing with peers, asking questions, teaching class content to others and many more. Another potential drawback from the lack of structured active learning opportunities is that course staff have no way to see the learning process of students and give feedback during this process.

One simple but direct way students can actively learn is by posing, and answering questions on the class material. However, a challenge that comes with this is generating enough content for students to fully practice. A potential solution to this problem is to utilize student-generated content. More specifically, students can attempt to create their own original questions related to the class material. The process of coming up with questions alone is already a valuable learning opportunity for students to engage with the course material. However, for these questions to be used for other students to answer, there are several problems that may arise, including the

difficulty in regulating the quality of these student-generated questions and providing timely feedback to students on their work. Feedback is especially important in helping students solidify and correct their understanding.

The cycle of lecture, homework and exam often observed in many courses is not only suboptimal for students but for the course staff as well. This structure limits students to showcase their understanding through just homework and exams. As a result, course staff have fewer opportunities to provide students with the feedback that is necessary and helpful for their learning. If more chances arise for students to submit their work for the course, such as student-generated questions, this would allow students to receive comments from staff on their understanding. This is also beneficial for staff, as staff can use this to gain a sense of how students are comprehending the material throughout the course. However, several problems may arise from this approach, mainly returning feedback to students quickly. Prompt feedback can be beneficial to students as later parts of the course are contingent on earlier material, and rectifying any misunderstandings sooner helps with better understanding of more complex concepts. Increase in student submissions lead to increased work for staff. In addition, returning helpful feedback requires more time per student submitted work. The workload and pressure for staff to provide helpful and prompt feedback increase when the course staff becomes the main provider of feedback.

In order to address these problems, we are proposing a new question generating workflow that is designed to benefit both students and staff. In this workflow, students create their own multiple-choice questions (MCQs) on class readings and answer questions written by their peers, students from previous semesters and staff. Additionally, they are also required to leave a comment for each question that they answer. When a student is done with writing their own question and playtesting other questions, their submission and comments go to staff for review. Once a student-written question is playtested enough by other students, staff can use those comments left by other student playtesters to determine the quality of the question and grade accordingly.

This proposed workflow was executed in multiple phases as part of MIT's Elements of Software Construction (6.031) course and finally resulted in Questionable,

a web application that handles both the students and staff needs for this workflow. Implementation of the workflow can be divided into two parts: the student-side and the staff-side. The beginning stages of the execution consisted of a simple Google form that allows students to submit their work and answer questions. Grading on the staff-side was maintained through a Google spreadsheet. Iterations of the designs and implementations led to the development of Questionable. Through the system, students are able to answer other student-written questions, leave comments on these questions and submit their questions. They are also able to keep track of the statuses of their past submissions and receive feedback from both staff and their peers who have answered their questions. On the staff side, Questionable provides a grading user interface for staff and presents all the relevant information in evaluating a student's submission. This includes the submitted question itself, the comments left by other students who have answered that particular question and their attempts at the question. These additional information helps the staff grade better and more quickly determine the effort put into these questions. This allows staff to manage increased submissions from students but still evaluate their work in a more efficient manner. Additionally, the feedback from both peers and staff reduces the pressure on staff to solely be in charge of giving back comments and the student author of the question can still receive good feedback.

Overall, the execution of this workflow within the context of 6.031 has achieved its initial goals of providing new learning opportunities for students and a better grading experience for staff. Students have been making efforts to create questions that are beyond just simple questions that require recollection of term definitions. Many of the student-generated questions turned out to be application type questions that require not only knowledge of the concepts but also a deeper understanding of the material in order to apply it to new scenarios.

Applying methods from item response theory (IRT) has also helped in estimating student abilities and measure two key characteristic of a question: the question difficulty and discriminability. While there are room for improvements in qualitatively analyzing questions, the results from IRT has provided a good initial framework to

analyze question quality in a qualitative manner. One of the goals from executing this new question generation workflow is to be able to determine good quality questions generated from students. Using the estimated difficulty and discriminability values from IRT, it may be possible to begin this filtering process by first eliminating questions of poor quality.

The overall grading experience for staff has improved as intended. While there is the change in rubric in how 6.031 is now grading student work generated from this workflow, the proposed system with Questionable has helped in alleviating some of the grading burdens for staff. In particular, the playtest results from other students and their feedback on the student-generated questions reduce the need to read through the student-generated question in detail to determine the overall question quality and the effort put into creating the question. Questionable provides a user interface for staff to make this grade decision easily by laying out the relevant information for grading.

This workflow of creating and evaluating questions, and receiving feedback from peers and staff can give students more opportunities to test their knowledge and to receive comments on their work without much additional effort from the staff. In the end, the goals of this new question generating workflow are to improve student's learning by increasing their engagement with the class material and to ease the grading process of these questions.

This new proposed workflow along with Questionable and the analysis done in this thesis make the following contributions:

1. Providing students with new ways to engage and practice with course material
2. Creating a system that allows for faster and increased feedback to students
3. Designing Questionable, a system which can handle both student and staff needs, especially by providing a better grading user interface for staff for more efficient grading
4. Investigating a way to analyze the question quality of student work using item response theory in conjunction with other metrics

Chapter 2

Related Work

2.1 Learning through Testing Effect

The proposed workflow is centered around the benefits of the testing effect, outlined in Roediger and Karpicke's study [12]. Learning material through test taking improves long-term retention. Even though repeatedly studying the material leads to better short-term recall rate, taking tests soon after studying leads to better performance on delayed tests that were given a week later. A big change in the proposed question generating workflow is that now students must also answer other questions written by their peers and staff. By taking these questions, students are taking test-like questions to solidify their understanding of the class material. Although Roediger and Karpicke's study observes the testing effect on one's retention and recalling ability, some of the questions created from the question generating workflow require students to go beyond recalling information and apply what they can remember. This new workflow can inspect how the testing effect affects students' abilities to answer higher-order questions.

2.2 Student generated content platforms: Codewrite, Peerwise, StudySieve

Unlike the traditional learning system where students learn mainly by studying given class material, some courses use platforms like Codewrite [7], Peerwise [5, 6, 10] and StudySieve [13] that facilitate the creation and exchange of student generated content for different types of courses. These courses incorporate student-generated content in class and introduce peer reviewing and peer feedback to improve student learning. As the name suggests, student-generated content refers to material, such as questions and notes, that students themselves create with the purpose of sharing it with their peers in class. In fact, the simple act of asking questions when learning science helps students to engage further with the concepts of the material and potentially launch them into more deep thinking [4, 13].

2.2.1 Codewrite: Sharing coding exercise

Incorporating CodeWrite into an introductory programming class has proven to help students to practice core aspects of the course more easily [7]. CodeWrite is a web-based tool that allows students to upload and share with their peers original coding exercise. Students can also write test cases for their coding exercises for others to run against. Additionally, students may publicly submit their solution to other coding exercises. Writing tests to test the correctness of code is often a challenge, especially for coding beginners, as there are many barriers to overcome when running test cases without errors. CodeWrite eliminates these obstacles and gives students a chance to focus on writing their original coding exercise and test cases. By the end of the semester, student-written exercises on CodeWrite collectively covered the set of Java language features the course aimed to cover. While CodeWrite creates an environment for students to actively practice the course objectives, those goals are based on skills that can be mastered through repetition and are related to some more basic programming knowledge. The proposed workflow, however, aims to im-

prove student learning through questions that go beyond basic coding and require higher-order thinking skills.

2.2.2 Peerwise: Sharing Multiple-Choice Questions (MCQs)

Similar to CodeWrite, PeerWise [5] is another web-based system that enables students to write, share and answer MCQ. Various courses that used PeerWise saw a correlation between the use of PeerWise and exam scores, especially among students who were initially at the bottom quartile of the class [10]. One use case of PeerWise was in a first-year programming course [5]. PeerWise was introduced mid-semester of this course after the first exam. Students were required to use PeerWise as part of their participation grades. The final exam scores show that there is a significant change in test grades among students who performed similarly on the first exam before PeerWise was introduced. In other words, students who were more active on PeerWise performed on average better than those who scored similarly on the first exam but were less active on PeerWise. Active PeerWise students who were initially at the bottom quartile of the class scored 10 more points on average than inactive students who were initially at the same quartile [10]. A student's activity on PeerWise can be categorized as the following: writing questions, answering questions written by peers, commenting on questions written by others. Of the different types of activities, the length of comments written rather than the number of questions answered seems to have positively affected student scores on non-MCQ exam questions. Additionally, the number of days active on PeerWise rather than the number of questions written had a greater effect on student exam scores. Both the questions in PeerWise and the questions used the proposed question generating workflow are MCQs and this study of PeerWise suggests that MCQs not only aid students to learn the material, but also, in some cases, improve their skills to answer more complex, non-MCQ questions [3, 10].

2.2.3 StudySieve: Free-response questions

Qualitative data from students on StudySieve, another web-based tool that helps students share their free-response questions, show positive learning experiences from students as well [13]. StudySieve differs from other student-generated content sharing tools in that after answering a question, the student is required to evaluate three things: the question, their answer to the question and an answer from another student. Students appreciated the variety of topics that the collection of questions covered and the different styles of questions they saw. Furthermore, students reported that seeing a range of correct questions helped them to learn. In particular, encountering incorrect questions made students cognizant of the common mistakes their peers made. This observation is crucial in the proposed workflow, as students will be seeing a mix of questions that have and have not been screened by staff. In fact, in the case of StudySieve, around 43% of the question explanations written by students were either missing or did not fully include an explanation for the answer choices [13]. Therefore, it will be expected that students will see some incorrect or poorly written questions during the question generating workflow. However, exposure to even these low quality questions can offer a good learning experience to students.

Studies of these student-generated content sharing platforms echo some common benefits in evaluating peers' work [7, 5, 6, 10, 13]. In some cases, many students left comments for the student author to see even if they were not required [5]. Denny et al. state that evaluating a question requires "application of higher-order cognitive skills," stimulating more participation from students. In other cases, evaluating their own questions and answers by other students helped students see which concepts are considered important and test-worthy [13]. In Hardy et al's study, lower quartile students seemed to have benefited from leaving comments on questions. Peer evaluation by having students leave comments on a question they have answered is another step in the new workflow, a crucial part that has the potential to be beneficial according to a study on PeerWise reports correlation between comment quality and test scores [10].

Unlike the student-generated content web systems like CodeReview, Peerwise and Study Sieve, the proposed workflow integrates staff grading in the feedback to students to avoid the common problems with peer grading. The currently student-generated content web systems mainly support interaction between students. This can become difficult for students when they are asked to grade work done by peers as a form of peer evaluation. Peer grading becomes a greater challenge when grading poorly written questions. As Bottemly and Denny point out, students are less confident about their opinions when they see incorrect questions [3]. Instead of explicitly stating that the question is incorrect, they merely express confusion in their evaluation of the question [12]. Furthermore, students tend to be biased when grading . They are especially stingy when grading the high-quality work from their peers. The proposed workflow hopes to explore the effect of staff involvement and see how it can improve situations like these. The workflow involves staff grading a student's question submission based on the quality of their original question and their effort in evaluating other questions. Without the training to develop the skills needed for grading, it is hard for peer-grading to be accurate [8]. By introducing staff feedback that can directly address possible problems with a submitted question, this proposed workflow hopes to see positive changes in how students are learning.

Studies of the use of tools like CodeWrite and PeerWise in a classroom reveal the effectiveness of peer review of student-generated content, but the skills and material used with these tools are different from the skills and material that are emphasized in more advanced programming classes . CodeWrite was used to help students in an introductory programming class to gain more practice in writing test cases, which is a skill that can be easily mastered by drill and practice [7]. Similarly, PeerWise was used mostly in introductory courses and the majority of questions in PeerWise were those that ask either to find or match the output of a code [6]. More advanced courses, however, delve into more conceptual ideas behind programming and deals with building bigger units of software unlike beginning programming courses. While the studies on CodeWrite and PeerWise have not observed how their systems work in non-introductory classes, the proposed workflow hopes to help students learn more

conceptual material and practice higher-order skills by incorporating peer review of student-generated content with staff feedback as well.

2.3 Item Response Theory (IRT)

Item response theory (IRT) is a psychometric model that is often used to evaluate the individual items that compose certain types of assessment of human behavior or performance [1]. In addition, it is used to also measure some latent quality of the subjects of these assessments. The theory is built on the idea that there is a relationship between the performance of the subjects of an assessment, an indirect indicator of some latent trait of the respondee, and properties of the individual items of an assessment. The properties of these individual items are called item parameters and the values of these parameters obtained from the IRT model allow for evaluation of the quality of the items. Unsurprisingly, IRT is often used in an educational context where it is used to evaluate questions of tests, especially standardized tests. The items in this case refer to the individual questions in the test. The latent trait of the respondees to be inferred is the intellectual ability of the test takers. Many item parameters measured by many IRT models are often the item discriminability and difficulty. The various item parameters that are measured may vary depending on the complexity of the different types of models within IRT.

IRT is not to be confused with classical test theory (CTT) in which the latter tries to characterize an entire test as a whole and the former focuses on the individual questions of a test. The IRT approach allows us to move away from the CTT assumption that all questions of a test are equally difficult, which in reality is not necessarily true. During the rest of the discussion of IRT, items will refer to questions of tests and the latent trait is the ability of the test takers.

2.3.1 Assumptions in IRT

IRT is built upon some crucial assumptions that define the relationship between the items and the subject's behavior. The first assumption is monotonicity of the ability

of the subject. This means that as the individual's ability is higher, the probability that they will answer a question item correctly increases. This relationship can also be visualized through a graph that maps the relationship between ability and the probability of getting the question correct as shown in Figure 2-1. As discussed further in Section 2.3.2, θ is used in the model to represent the ability of the individual where a low θ value represents low ability and an increasing θ indicates a higher ability. The shape of the graph shows that probability of getting a correct never decreases with an increasing θ . The second assumption is unidimensionality in which the responses to the items are measuring just one latent trait of the respondees, and that trait is what drives the respondee's answers to the items. Another core foundation of IRT is the idea of local independence of response. More specifically, how an individual responds to a particular item is independent from their response to another item on the test. Finally, a crucial characteristic of IRT, one that distinguishes it from CTT, is the invariance of the item parameters to the group of individuals. Item parameters are not affected by the individuals. The parameters are measures of characteristics of the item that is inherent to the item. This implies that given the responses from any group of individuals, the item parameters should not change.

2.3.2 Item Response Function (IRF)

The relationship between the item parameters and the ability of individuals that the IRT model can be described as a probability function where the input is the ability of an individual, represented by θ and the output is the probability that an individual with that ability will answer the question correctly. The item response function (IRF) is the formal name of this function and considers both the item parameters and the ability in its computation of the probability. The IRF varies slightly model-by-model but differs mainly in the number of item parameters the model considers. For an IRT model that considers two item parameters (two parameter logistic model, 2PL), the IRF is the following:

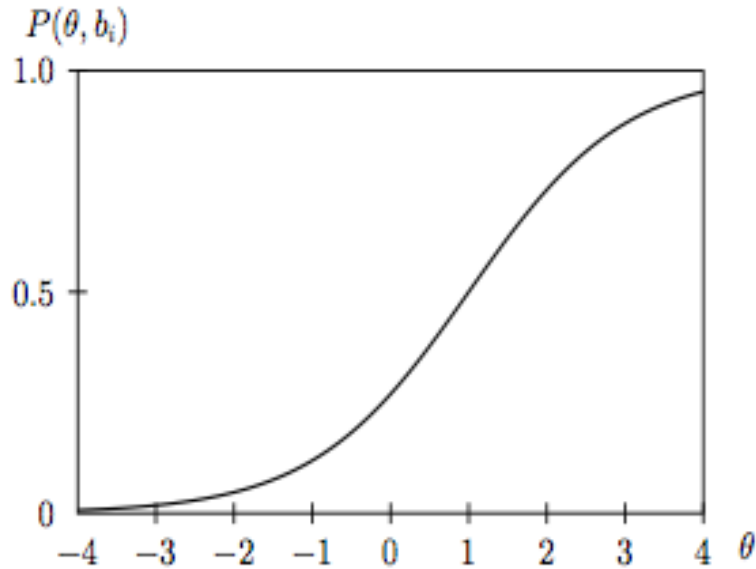


Figure 2-1: Item Characteristic Curve (<https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory>)

$$P(\theta) = \frac{a * \exp(\theta - b)}{1 + a * \exp(\theta - b)} \quad (2.1)$$

where:

θ = the ability of an individual

a = discriminating item parameter

b = difficulty item parameter

Graphing the IRF of a 2PL model yields the Item Characteristic Curve (ICC) as shown in Figure 2-1. The ICC is a S-curved, which shows the monotonicity assumption as described in section 2.3.1.

Item Parameters

Item parameters are representative of the characteristics of the item we want to measure. They are important not only in the computation of the probabilities in the IRF but also provide visual insights and the following are main item parameters included in many IRT models.

Item Discrimination (a) This parameter, noted as a determines how well an

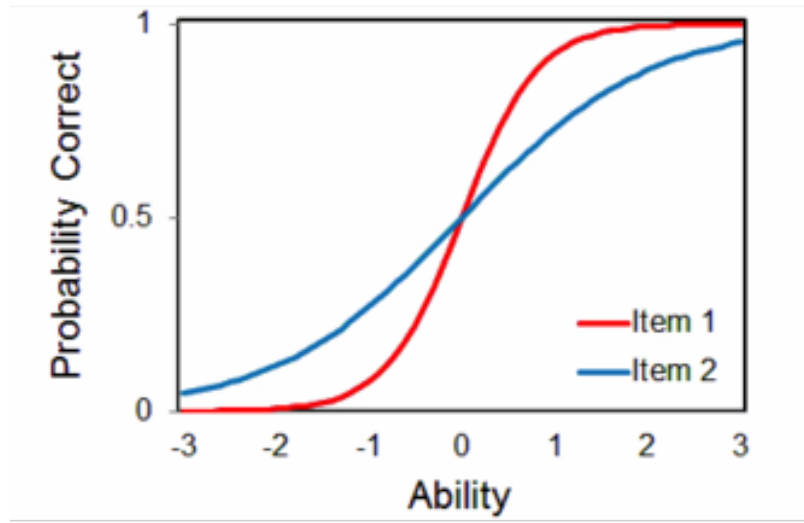


Figure 2-2: ICCs with different discriminability. Item 1 has a steeper slope (higher a value) than Item 2. (<https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory>)

item is discriminating against individuals with higher ability (high θ) compared to those with lower ability. It can also be thought as the rate at which the probability that a respondee answers a question correctly changes as the ability changes [1]. This value can be determined by looking at the slope of an ICC curve at the inflection point. As shown in Figure 2-2, a steeper slope, which translates to higher discriminability, can better distinguish individuals with different θ s by increasing the probability of correctly answering for higher θ s and lowering that probability for lower θ s. While theoretically, a can take on any value, this should usually be non-negative. Negative a values mean that the item cannot discriminate individuals in different ability groups properly.

Item Difficulty (b) The difficulty parameter, marked as b , represents the difficulty of an item where the notion of hardness is defined as the probability for an individual to answer correctly is 0.5 given an ability level. Figure 2-3 shows the ICC of three items, all of which have the same a value and only differ in their b values. This shows how changes in b result in horizontal shifts across the graph. With higher

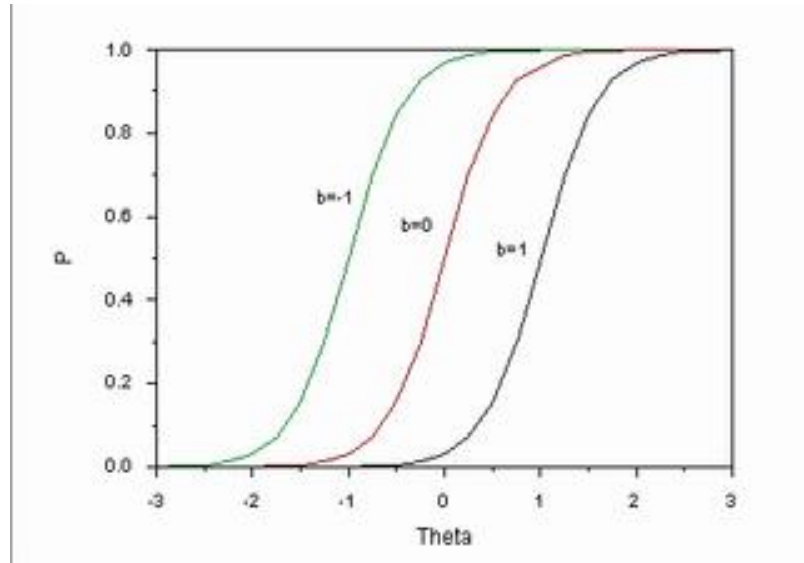


Figure 2-3: ICCs with different difficulty values of -1, 0 and 1, respectively, from left to right. (<https://www.mailman.columbia.edu/research/population-health-methods/item-response-theory>)

b values, the ICC shifts to the right and as a result for given a θ , the probability of getting the question correct decreases compared to when $b = 0$.

Hanson describes in his paper how IRT can be applied to an observed data of student responses to questions. It goes into detail of how this data can be used to predict both item parameters and abilities of the item respondees [9]. IRT would be a useful tool in analyzing what could be an subjective topic, like the quality of a question, in a more systematic and qualitative way that is highly relevant to the goal of evaluating questions. MCQs are good questions for IRT, as determining whether a response is correct or incorrect is a binary outcome. Furthermore, MCQs are used not only in the proposed workflow but also in many classes in general.

Chapter 3

Design & Implementation

3.1 Proposed New Question Workflow

The way students receive feedback in a traditional classroom setting yields many inefficiencies for students and staff. In a usual class, the most common procedure for students to receive feedback is to take some assessment or complete assignments for staff to grade. The staff usually receives a bulk of submissions at once and must go through all of the submissions to grade. Once the grades are returned to students, only then can students receive some comment on their work. The nature of this type of feedback system usually creates a cycle involving students and staff, and this cycle is slowed down by the imbalance of the number of items that must be graded and the number of staff who are able to grade. The delay in receiving feedback is not beneficial to students either, as this is not only untimely but also leads to fewer opportunities for students to reflect upon their work during the course of the class.

In this thesis, we propose a new workflow in the way students can generate content related to class material and receive feedback. This proposed workflow takes a different approach from the traditional way that aims to alleviate some of the issues pointed out earlier. Additionally, this workflow focuses specifically on multiple-choice questions (MCQs) written by students on the material they have learned. The steps of proposed workflow are as follows:

1. Students first come up with their own original MCQ and write an explanation for

each of the answer choices.

2. Before submitting their questions, students answer and leave feedback on questions that written by either their peers in the class or a staff member of the course.
3. Both the original question written by the student and their answers and feedback to other questions are recorded and sent to staff for review.
4. Staff reviews questions submitted by students as well as how other students have answered the question in review.
5. Once staff is done grading, both the staff and peer feedback are delivered back to the original student author.

3.1.1 Student-side changes

Answering and commenting on peer-generated question

Other than writing an original question, the biggest change for students in this proposed workflow is answering and evaluating questions written by other members of the course. This can be other students currently enrolled in the class, students from previous semesters of the course or course staff. While the act of thinking and creating original content is a good way to further engage with the class material, through this workflow, students get the extra opportunity to test their understanding of the class material in a low-stake environment. This allows for testing effect to come into play and solidify their understanding of the class material [12]. Furthermore, peer evaluation has shown to engage higher-order of cognitive skills [5].

By incorporating peer evaluation into the loop, this proposed workflow also addresses the issue of the lack of feedback students receive. Students are now able to get feedback on their work from multiple people. This combined effort from various people increases the chance for students to hear back on various aspects of their work. Such structure allows greater and more creative involvement with the course material for students without heavily increasing the work load nor skewing the work balance of any member of the course.

3.1.2 Staff-side changes

The time it takes for course staff to grade is a bottleneck in the traditional feedback cycle and hence slows down the time it takes for students to hear back about their work. However, through the introduction of peer evaluation, the workflow allows grading to become a more efficient task. Usually, course staff is the only provider of feedback and most of the time, the only member of the classroom to grade a student work. Furthermore, they are the first to evaluate a student's work, meaning more effort and time are needed to assess student's work. However, through this workflow, other students can now partake in the evaluation step by answering questions and leaving feedback. Knowing how other students have responded to the question, staff can use this data to better and more quickly gauge the quality of a question. Additionally, the pressure to leave thorough comments for the student author of a question decreases as other students are now providers of feedback.

3.1.3 Application of Question Workflow in 6.031

This proposed question generation workflow is now integrated into MIT's Elements of Software Construction (6.031) course's existing nanoquiz makeup procedure. In 6.031, nanoquizzes are 3-minute-long quizzes given in the beginning of class to test students on the reading assigned for that class. These nanoquizzes altogether account for around 10% of the final grade. After each nanoquiz, students have the option to submit a makeup for a nanoquiz to gain back some of the lost points. This acts as an incentive for student submissions. The existing makeup procedure is simple. Students write their own question for that class reading and submit it for staff to grade. When submitting a makeup students must not only include the question they have written but also the answers and explanations for those answers. These nanoquiz makeups can earn up to 0, 1/2 of the lost points back depending on how well written the submitted question and the answer explanations are.

The nanoquiz makeup procedure for 6.031 has been modified to reflect the proposed workflow starting from the fall semester of 2019. The workflow involves two

main entities, namely students and staff, and can therefore be broken down into two parts: the makeup submissions for students and the makeup grading for the teaching staff. Each part was introduced into the course in increments and independent of each other.

3.2 Questionable: web application for question workflow

Questionable is a web application that is able to handle all interactions and actions that occur in the proposed question generation workflow. Through Questionable, students can submit their work, answer and leave feedback on peer-written questions, maintain all their submissions and receive feedback about their work. For course staff, this web app increases the ease of grading by providing a grading interface that combines useful signals from student's work, and displays these signals in a way that is easy to process only the appropriate information used for determining a student's work. Furthermore, it provides structure for staff to maintain and keep track of the grading progress.

Although Questionable was created and developed within the context of 6.031 and uses tool specific to the course, the general structure of the user interface is generalizable and can be used for other courses that may adopt this new form of student question generation. The rest of the thesis shows the usages of Questionable, some of which are specific to 6.031.

3.2.1 Overall Structure of Questionable

Questionable is a Node.js web application written in HTML, CSS and Javascript. No frontend framework was needed through the development process, as most of the frontend components did not require the complexity that called for such frameworks. The backend consists of MongoDB as the database used in conjunction with Mongoose. The database stores questions written by students and staff from the current and

previous semester. It also stores playtest results of students answering other questions, including all the attempts made while answering questions and comments left for those questions.

3.2.2 Design of Student-side UI

The main functionalities needed for students of Questionable are writing their own questions, submitting their questions, answering questions written by others and viewing all of their submitted questions. In order to ensure that the questions written by students are getting reviewed by peers, the workflow includes answering other questions as part the makeup submission process. This means that of the core functionalities needed, all but one happen as a series of event. In order to make a practical UI that is suitable with the submission process, the student-side UI is a simple three-page form. The first page allows students to view previously submitted makeups, if any. The second page is where students answer peer-written or staff-written questions and the last page is where they finally get to submit their own question.

Initial Page on Student-side

Questionable is currently accessible to the members of the MIT community and requires a certificate in order to login for security reasons. Figure 3-1 shows the first page a student would see when logging into Questionable. If they have previously submitted a makeup, the table on the top lists them and the status of each submission. A status can be blank, accepted or rejected. A blank status means that the makeup is not graded yet. Accepted means that the makeup is graded and is getting back points whereas a rejected makeup is a graded makeup that is not good enough to gain back lost points. Students may select the class for which they wish to submit a makeup as shown in the bottom half of Figure 3-1. The three circles at the bottom of Figure 3-1 indicate which stage of the form the current page is at.

6.031 Makeups

Submitted Makeups

Below are the makeups you have submitted so far. Note that you are allowed only one submission per class.

Makeup submissions are not graded in order, so if some of your submitted makeups have not been graded yet, please be patient -- you will see the statuses of your makeups below eventually.

Class	Status
01: Static Checking	accepted
02: Basic Java	accepted
03: Testing	
14: Recursion	
17: Grammars	

Please select a class

Choose the class you want to submit a make up for. Only classes whose grades were posted within the last week (7 * 24 hours) can be made up.

Next



Figure 3-1: Page 1 of the makeup form in Questionable for Students

6.031 Makeups

First Answer These Questions

Here are some questions that were previously written for this topic -- possibly by course staff, possibly by other students. Please answer each one, read its explanation, and then comment on the question in the textbox below it. Your comments may be shared with the question's author, so please follow the same norms as in [code reviewing](#).

Question 1

Which of the following are required methods to be implemented of the Music interface?

- `concat(m1, m2)`
- `play(player, atBeat)`
- `duration(m1)`
- `transpose()`

CHECK

Please comment on your experience with question 1, as if you were talking to the question author: Was it easy, hard, or impossible to answer? Are there any confusions or mistakes in the question, its answer, or its explanation?

Enter your comment about the question here.

Question 2

Which of the following are true of a composite pattern?

- Separate categories for single objects and groups of objects.
- Tree data structures that have primitives at the leaves and composites at the other nodes.
- In Music as described in the reading, Note, Rest, and Concat are all part of a composite pattern.
- All recursive data types define a composite pattern.

CHECK

Please comment on your experience with question 2, as if you were talking to the question author: Was it easy, hard, or impossible to answer? Are there any confusions or mistakes in the question, its answer, or its explanation?

Enter your comment about the question here.

Question 3

Which of the following choices would fully define idea of a Composite Pattern?

- Primitives are leaf nodes
- Creates a tree data structure
- Composites are internal nodes

CHECK

Please comment on your experience with question 3, as if you were talking to the question author: Was it easy, hard, or impossible to answer? Are there any confusions or mistakes in the question, its answer, or its explanation?

Enter your comment about the question here.

Previous

Next



Figure 3-2: Page 2 of the makeup form in Questionable for Students

Evaluation of Peer-written questions

After selecting the appropriate class, students may then advance to the second part of the form to answer peer-written questions. The system queries from the database three questions submitted for that class. These questions are either written by students from the current or previous semesters or by course staff. The three selected questions are usually one question from a previous semester and two from the current semester. The reason questions from previous semesters are included is that these questions are already graded and have been checked at least once for its quality. While studies on sharing peer-generated content have shown that there are benefits for students to evaluate even poorly written questions [6], the hope is to still provide students with the opportunity to also learn by answering acceptable questions that have been verified for quality. In the case where there are no questions from the current semester to pick from, then all questions are simply questions from previous semesters.

During this second part of the form, students are answering questions and leaving feedback for the original student author of those questions, if applicable. Every time a student attempts to answer a question by clicking on the "Check" button, they can see whether they answered the question correctly or not. Students must attempt to answer the question at least once in order to view the correct answer choices and the provided explanation, which they may do so by clicking on the "Explain" button. Figure 3-3 is an example of someone checking the answer after answering the question incorrectly. If a student answers a question correctly without clicking on "Explain" to view the answers, the question automatically renders the answer and explanation as shown in Figure 3-4.

Upon answering questions, students must also leave a feedback by commenting on their experience with the question. Other peer-generated content sharing platforms have employed various ways of feedback. Most of them used some combination of scaled ratings and comments [3, 6]. In the case of PeerWise, around 32% of the students found comments useful. The platforms that used the rating systems used

Question 1

Assume we have the following chunk of code:

```
Music r = rest(1);
Pitch p = new Pitch(A).transpose(6);
Music n = note(1, p, GLOCKENSPIEL);
List<Music> s = List.of(r, n);
```

Which of the following is a valid Music?

- p
- concat(r, r)
- concat(p, p)
- s

▶ A Pitch is not a Music (answer p), nor is a List<Music> a Music (answer s). concat(r, r) and concat(p, p) are both valid Music objects.

CHECK EXPLAIN

Please comment on your experience with question 1, as if you were talking to the question author: Was it easy, hard, or impossible to answer? Are there any confusions or mistakes in the question, its answer, or its explanation?

good question. Directly related to reading for class

Figure 3-3: Answers and explanation after not answering question incorrectly

Question 3

Which of the following are true of a composite pattern?

- Separate categories for single objects and groups of objects.
- Tree data structures that have primitives at the leaves and composites at the other nodes.
- In Music as described in the reading, Note, Rest, and Concat are all part of a composite pattern.
- All recursive data types define a composite pattern.

▶ Composite patterns treat single objects and groups of objects the same. A tree structure with primitives at the leaves and composites at the internal nodes ensures that both single objects and groups of objects are treated the same. The three objects listed in the third answer are all part of a composite pattern. Recursive data types alone don't guarantee that the objects follow a composite pattern.

CHECK EXPLAIN

Please comment on your experience with question 3, as if you were talking to the question author: Was it easy, hard, or impossible to answer? Are there any confusions or mistakes in the question, its answer, or its explanation?

Enter your comment about the question here.

Figure 3-4: Feedback generated when student answers question correctly

ratings as part of student grades or as a means for students to filter questions to answer. However, within the context of this workflow, the impact that the rating system would have would not be meaningful. As a result, we decided to just have comments as the main way for students to leave feedback.

Writing and Submitting Original Question

The final step of the workflow for students is to create their own question for submission as shown in Figure 3-5. The top section outlines the expectations from staff regarding what the question should satisfy in terms of content and format. The middle area is an editable textbox where students may type in their question. The two

6.031 Makeups

Then Write Your Own Question

Now write your own question related to this topic.

- Your question must be **multiple-choice**, either "choose one answer" or "choose all good answers."
- Your submission must **encode the correct answers** and **provide an explanation** of why each choice is correct or incorrect. The Answer Preview box below should have the correct answers checked and the explanation visible.
- Your question must be **relevant** to the reading for the class you're making up.
- Your question should be **clearly answerable** for someone who did the reading, but hard to get right for someone who didn't. Trick questions and trivial questions will not receive credit.
- Your question should be of **similar length and complexity** to reading exercises or nanoquiz questions.
- Your question should be **original**. Copying text directly from the reading, or making a small change to an existing reading exercise, will not receive credit. Copying from student-written questions, like the other questions you just answered on this form, violates the [collaboration policy](#).
- Your makeup question may be playtested anonymously by other students in the class this semester, and accepted makeup questions may be used anonymously as ungraded review material in this or future semesters.

Your Question+Answers

Submit your multiple-choice question here, writing it in [handx format](#).

Here is the text of my question. Use backquotes for `code` style, for example to talk about `String` or `someVariableName`.
For multi-line code, indent each line 4 spaces:

```
while (true) {  
    b = a * b;  
}
```

How should we format the answer choices?

[] Incorrect choices have an empty box of square brackets

[X] Correct choices have a box with an "x"

[X] You can write a choose-one or choose-all question

> Write the answer explanation on lines

> starting with ">".

>

> Be sure to explain why each option is correct

> or incorrect.

Question Preview

Here is the text of my question. Use backquotes for `code` style, for example to talk about `String` or `someVariableName`. For multi-line code, indent each line 4 spaces:

```
while (true){  
    b = a * b;  
}
```

How should we format the answer choices?

Incorrect choices have an empty box of square brackets

Correct choices have a box with an "x"

You can write a choose-one or choose-all question


Answer Preview


Here is the text of my question. Use backquotes for `code` style, for example to talk about `String` or `someVariableName`. For multi-line code, indent each line 4 spaces:

```
while (true){  
    b = a * b;  
}
```

How should we format the answer choices?

Incorrect choices have an empty box of square brackets

Correct choices have a box with an "x" 

You can write a choose-one or choose-all question 

> Write the answer explanation on lines starting with ">".
Be sure to explain why each option is correct or incorrect.

Figure 3-5: Question submission page of student-side of Questionable

purple boxes are previews of content of the textbox above and update when any part of the text in the textbox changes. The left preview labeled "Question Preview" shows the question as one would see when trying to answer the question. The preview on the right labeled "Answer Preview" is the question in its expanded form that shows the correct answer choices and the explanation.

3.2.3 Student-side Implementation

The selection of the questions for the student to answer is handled by the server to diversify the questions the student can answer. When a student selects a class they wish to submit a makeup for, a request is sent to the server which queries the database for questions from the past and current semester related to the reading for that particular class. Questions that were submitted for makeups from the fall 2019 semester onward have a count of the total number of playtests stored in the database. The system aims to pick three questions to playtest: two from the current semester and one from previous semesters. Any random question from the previous semester is selected for playtest. For picking questions from the current semester, the number of total playtests is considered by prioritizing questions that have fewer playtests. Doing so can evenly distribute the number of students answering a question.

As a student is working through the three questions to answer, the system also keeps track of the attempts they make. As shown in Figure 3-2, all questions students answer and create are rendered in that format. It is rendered using a tool called HandX that is used within 6.031. HandX takes a text written in Markdown format to display questions that can be answered in an interactive way. Questions rendered through HandX are formatted so that a user must answer a question by clicking the "Check" button. Only after they press "Check" button do they have the option of viewing the answers and explanation. A user can check their answer for any number of times until they get the question correct, at which point the explanation appears and disables the "Check" button. We consider every attempt as every time a user clicks the "Check" button. A click event listener is attached to the button which sends a POST request to the server recording the checked and unchecked answer choices at the moment when the user clicked "Check". With this information, it is possible to see which of the answer choices they have answered correctly. This is useful later on when evaluating the quality of a question and the effort a student has put in when answering a question, which is a factor of their makeup submission grade.

There is form validation on the makeup form in order to ensure that components

that are necessary and important to the workflow are there. First, the system checks that the student has left comments for all three questions they have answered. If they click on the "Next" button on the second page shown in Figure 3-2, the page will let the student know if they did not leave comments, highlight the unfilled areas and ask students to write their feedback. Lastly, the form makes sure that all checkboxes on the bottom of page 3 (Figure 3-5) are filled out before submitting. This is to encourage correct formatting of the written questions. As mentioned before, the questions rendered on the form and Questionable in general follow a format based on Markdown and when certain characters are off that heavily impacts the rendered output of the question. Examples of common incorrect displays are questions with the answers being exposed, code blocks not being shown correctly, regular text appearing within code blocks when they should not be and many more. Errors like these have the potential of negatively impacting a student's experience in answering these questions and distracting them from focusing on the question content itself.

3.2.4 Staff-side UI

Questionable has a staff-side user interface which is used mainly to grade and manage makeup submissions. The biggest design challenge for the grading interface was in coming up with a layout that contains useful information for evaluating a question and that displays these information in an user-friendly manner that allows the staff grader to grade efficiently.

Staff members are allowed to view all the makeup submissions and Figure 3-6 shows the rendered view of these submissions. It is essentially a table of all the questions listed by class and by order of submission. The search bar on the top allows to search for any substring that occurs in any of the entries. Each row contains only the minimum information that summarizes the status of each submission. One thing to note is the "# of Playtests" entry on the far right. This indicates the number of students who have answered and playtested the submitted question. Questionable tries to spread out the number of playtests for each submission for a class where the maximum number of playtests is capped at three. As a result, staff do not grade

submissions until a submission has been playtested by other students at least two or three times. This also means that makeups that are submitted later are not be playtested as much. These makeups show up at the end of the table and staff simply grade those despite low playtests count.

6.031 Makeup Grading Overview

In 6.031, makeup submissions are either accepted or rejected. If accepted, the student receives 1/2 points of the lost points from the nanoquiz and if rejected, they get no points back. When reporting back to students on their submitted makeup, we simply indicate whether their makeup was accepted or reject. On the staff-side, however, grades are broken down further into three categories of different names: check, borderline, minus. Minus is equivalent to a "rejected" makeup on the student side whereas check and borderline are "accepted". These grades are determined based on the effort. More specifically, staff looks for evidence of effort put into the makeup as a whole. This means effort in the question submitted and the playtest feedback students give to their peers. Check grades are awarded to makeups that fully satisfy the staff's expectation. The borderline grade is usually given when a submission is lacking effort in either their written question or their attempt at answering other questions. Borderline grades are given first if a makeup is not completely satisfactory with some warning to the student about the lack of effort in their makeup. Repeated series of borderline grades will eventually lead to a minus grade the next time their submission is not good enough for a check.

Grading User Interface for Staff

The grading UI for staff is aimed to reduce the grading time and effort for staff. In fact, one of the motivations of this proposed workflow is to unload the grading burden of the staff by distributing some of the evaluation task to the students answering the question. This means that more can be observed student behavior and more information can be collected regarding a makeup. As mentioned before, grades are effort-based where effort is evaluated in mainly two places: the submitted question

Makeups						Dashboard
spec						
Class	Student Name	Kerberos	Grade	Grader	# of Playtests	
06: Specifications	Jane Doe	jdoe	check	alyssap	2	
06: Specifications	John Smith	jsmith	check	alyssap	2	
06: Specifications	Amelia Brown	ameliab	check	alyssap	2	
06: Specifications	Jeff Wilson	jeffw	check	alyssap	2	
06: Specifications	Lily Kim	lkim	check	alyssap	1	
06: Specifications	Eric Baker	bakere	check	alyssap	0	
07: Designing Specs	Amanda Cooper	acooper	check	bitdiddle	3	
07: Designing Specs	Luke Young	lyoung	borderline	bitdiddle	3	
07: Designing Specs	Ava Anderson	andersonv	check	bitdiddle	3	
07: Designing Specs	Jeff Wilson	jeffw	check	bitdiddle	2	

Figure 3-6: List of all makeups on staff-side of Questionable

Makeups		Dashboard
<p>< Amanda Cooper (acooper) 07: Designing Specs (0 ungraded, 24 graded) submitted 7 days before deadline</p> <p><small>graded by: bitdiddle</small></p>		
<p>Comment to student:</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p>✓</p> <p>↑</p> <p>↓</p> <p>-</p> </div> <p>History: ✓</p>	<p>Note to staff:</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;"> <p>This is a restatement of a core idea from the reading. Note that, in the future, questions should be more original, or they will not receive credit.</p> </div>	
<p>Question Source</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p>Designing Specs</p> <p>Which of the following creates a stronger spec?</p> <ul style="list-style-type: none"> <input type="checkbox"/> A weaker precondition with the same postcondition <input type="checkbox"/> A stronger precondition with the same postcondition <input type="checkbox"/> The same precondition with a stronger postcondition <input type="checkbox"/> The same precondition with a weaker postcondition <p>▶ A spec is stronger if its postcondition is stronger in relation to its precondition. The only options that accomplish this are the first and third options.</p> <p>CHECK</p> </div>	<p>Playtests (2)</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p>This question was easy and straightforward to answer. -ameliab</p> <p>Question 6&mp; explanation easy to understand. -lyoung</p> </div>	<p>Submitter's Answers to Other Questions</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p>Question 1 Change "number" to "index" (was confusing)</p> <p>Question 2 Was easy</p> <p>Question 4 Was different than other questions - which is good!</p> </div>

Figure 3-7: List of all makeups on staff-side of Questionable

and their attempt at the other questions they answered. Questionable keeps track of several things that allow observations of effort in these two parts.

Questionable not only saves the comments students leave, but it also keeps track of every attempt a student makes when answering a question. Since all of the makeup questions in 6.031 are MCQs, it is easy to determine whether a student answered correctly. An attempt is considered correct only when all answer choices are appropriately selected or deselected. For a submitted question, it is possible to deduce the quality of a question by observing how others have answered the question and the progression of their attempts. If a question where most of the playtesters have attempted many times or where many were not able to eventually answer correctly, then that might be a signal of a bad question. On the other hand, a question where most playtesters can answer correctly within a couple of tries could be a better indicator of a well-formed question. A history of student attempts at a question also demonstrates the effort put into that question. If a student simply answers a question once incorrectly and does not attempt to try again, then that could very well signify low effort in actually trying to answer the question.

Comments also play a role in determining the quality of a question and the effort a student puts into playtesting other questions. Overall positive comments from playtesters are good signs of the question being appropriate. However, comments that point out major flaws in the question or where majority of the comments echo the same problem can indicate that the question will most likely be a poorly written question. If comments from a single student are mostly very short and same for the questions they have answered, then it is a clear indicator that the student did not put in the time to engage with the peer-written questions and to give feedback.

The Questionable grading interface brings in all these attempt histories and comments, as well as other useful signals, and lays them out in an intuitive way that helps the staff grader to determine a grade for a given makeup. Figure 3-7 shows an instance of a graded submission. The top header lists the name and username of the student submitter, the staff grader of the makeup, if any, and the grading progress for that class. Graders are able to navigate between submissions for a given class by

clicking the left and right arrows found in the header of the page. In 6.031, two staff members are responsible for grading submissions that come in for a single class and the grading is divided between the two. As a staff is grading through submissions, they can see how many have been graded and ungraded and from that can determine when to stop or continue grading. The upper portion of the page is dedicated for the staff grader to leave grades, comments to students or notes to staff regarding the submission. Grade related aspects, such as the actual grade for the submission and the previous grade history for a student is grouped together on the top left. The three possible grades are buttons and a grader assigns grader by selecting one of those buttons. The buttons shown on the left side of Figure 3-7 correspond to check, borderline, minus from top to bottom, respectively. The "History" section found below the buttons lists all the previous grades the student submitter received up until that submitted makeup in chronological order. This section is useful in determining between a borderline and minus. As is the case in Figure 3-7, the student was given a borderline, but not a minus as they have not received a borderline nor minus before.

The bottom half of the grading page lays out the submitted question, the results of playtests made by the student as well as playtests done by other students in answering this submitted question. The left column of the bottom half section shows the rendered question in its expanded form revealing the correct answer choices and explanation. The tab named "Source" found above the rendered question is the plain text of the question the staff grader may read if the submitted question was formatted incorrectly and resulted in an unreadable rendered question. Since other playtests of this question help determine the question quality and the effort put into this writing this question, these playtest results are placed right next to the question as the center column labeled "Playtests". The comments shown under this column show only the necessary information: the comment and the attempt history made by each playtester. The attempt history is the column of squares. Each attempt is grouped vertically. One attempt would be a single column, and two attempts would be two columns and so on. A square can either be gray or filled with a green checkbox. Green checkbox represents a correctly marked answer choice. In other words, a column with

all green checkboxes will indicate a correct attempt. Attempts progression are shown left to right, with the right-most column being the last attempt. Ideally, an attempt history would end with a column of green checkboxes.

The rightmost column labeled "Submitter's Answers to Other Questions" is meant to show the student's attempts in answering other questions. The column contains links to the questions the student answered, the comments they have left and the attempt history for each question. Attempt history that are long are condensed to show only the first and last two, as shown in the first attempt history in the "Submitter's Answers to Other Questions" column in Figure 3-7.

3.2.5 Staff-side Implementation

When rendering the grading UI, the system only shows playtest attempts that were done as part of a student's makeup. The setup of the three-page form on the student side and how Questionable keeps track of all attempts lead to cases where students may answer questions from the second page of the makeup form but never actually complete the makeup submission process by not submitting a written question on page 3 of the makeup form. When grading a student's work, only including playtest results that were done as part of a complete submission seemed appropriate since it is clear that those attempts were done with the full intention of submitting a makeup for a grade. As a result, the system searches for all the playtests done in that semester, looks up the associated user of the playtest and filters by checking if that user has submitted a makeup for that class.

A security issue with rendering inputs from students is that there is the danger of cross-site scripting (XSS) attacks. Measures are taken by escaping the questions and comments written by students upon rendering. The appropriate div elements are first created outside of the DOM and then added to the DOM afterwards. Additionally, all pages of Questionable uses the cross-origin resource sharing (CORS) policy as an extra layer of security to prevent any resource's origin that is not 'self' from executing.

All activity made by the grader is saved automatically and the grader does not have to go through any additional action to save their grading progress. The moment

a grader assigns a grade or leaves a comment for student or staff, the system marks that staff as the grader and stores the grade and comments, if any. When a grader begins typing into the textbox to leave a comment, the page waits for the typing to stop for a few seconds and then a POST request is sent to the server to save the comment to the database.

Chapter 4

Evaluation

The main objectives in introducing this new question generation workflow is to foster a cycle where students can engage more actively with the class material and for staff to be able provide feedback in a more efficient way. Another goal is to have good quality questions generated from students. These three things can be evaluated both directly and indirectly by analyzing the various outcomes as a result of this workflow.

Level of engagement from students could be observed by looking at the efforts that are put into the question and the playtest attempts. The comments that student leave can also give insight about how engaged students have been during this workflow process. Looking at comment length can also be an indirect indicator of how involved students are in this learning opportunity. A successful exchange of learning from student generated content may involve insightful and long comments.

Evaluating whether the grading process improved for staff can be done both quantitatively and qualitatively. Comparing the time spent on grading can be a measure of how grading experience has changed. Surveys from the staff members also provide firsthand information of how their grading experiences have been with the new workflow and with Questionable's grading interface.

Assumptions of item response theory, such as invariance and unidimensionality described in Section 2.3.1, allow us to apply item response theory to the various data from 6.031 to infer characteristics like difficulty and discriminability of student-written questions.

Other metrics calculated from the playtest data are also examined as part of the evaluation process.

4.1 Application of IRT on 6.031 Data

IRT can be applied to estimate item parameters and student abilities from the response data of students to various questions [2, 9, 11]. For the evaluation process, the 2-parameter logistic (2PL) model in IRT was used, as it considers the effect of just two aspects of a question: difficulty and discriminability. Given the 2PL model and assuming that students ability have a normal distribution, it is possible to estimate simultaneously the item parameters for questions and the abilities of students from the observed data that captures how students answer questions using maximum marginal likelihood expectation (MMLE) and expectation-maximization (EM) [2, 11].

Student responses to nanoquizzes in 6.031, as described in section 3.1.3, were used to estimate student abilities and the two item parameters (difficulty and discriminability). Using the nanoquiz data seemed reasonable, as the nature of the nanoquizzes seemed to be most similar to that of the questions that are written by students. Many of the nanoquiz questions are MCQs and the content of the nanoquiz and student questions submitted through Questionable are both based on the class readings. In fact, the study from Peerwise shows that the question style submitted by students on Peerwise is similar to the style of test questions from the course [6]. Furthermore, these nanoquiz responses are collected under a controlled environment whereas other data from 6.031, mainly reading exercises embedded in class readings, has the potential to contain a lot of noise in the data. Therefore, the abilities of students estimated from the nanoquiz data would ideally be representative of a student's ability in answering nanoquiz-like questions.

The invariance assumption allows us to take the student abilities estimation from the nanoquiz data and use those values to infer the item parameters of the questions student answers, particularly the student-written questions [9]. This approach to estimate the item parameters of the student written question works should be valid

in theory because the values of the item parameters are not dependent on the abilities of the respondents of the question [2].

There are certain limitations when applying IRT using the maximum likelihood and this requires making adjustments to the process that affect the result. The first problem is that there is no unique metric for the ability scale and requires anchoring the scale to a determined midpoint. Many implementations of IRT using some variation of maximum likelihood usually assume a unit normal distribution where the midpoint is anchored to 0. While there are several methods of estimating the abilities and item parameters in IRT, the IRT implementation outlined in Hanson's paper uses an iterative process using an expectation-maximization (EM) algorithm [9]. In Hanson's implementation, one important modification is that the ability parameter is treated as a discrete variable, rather than continuous. This is to make integrating over the ability values computationally feasible. Hanson's paper assumes that there is a complete data for all respondents. However, for the actual nanoquizzes, there are situations where a student does not attend class and ends up not taking a nanoquiz. In our implementation of IRT, we handle this by just using the responses from students who were actually present for that nanoquiz when estimating the parameters and student abilities.

4.1.1 Processing Student Response Data

The IRT model that was used was a 2PL dichotomous model and the student responses were processed accordingly. A dichotomous IRT model means that a student response is binary. It is either correct and incorrect. The questions on the nanoquizzes in 6.031 are mostly MCQs but sometimes are free response questions. In processing MCQs, we defined the unit of a single item, as used in the context of IRT, as a single answer choice of each MCQ on the nanoquiz. In other words, if a MCQ had four possible choices, this would actually be processed as four separate items in IRT. For each item, if the student had correctly selected or not selected that answer choice, then the student is considered to have gotten that item corrected. As for free response questions, the question itself is considered a single item and the correctness of the

student responses is determined by staff.

4.2 IRT Results on 6.031 Nanoquizzes

4.2.1 Ability (θ) Estimation Results

The ability of the respondents, noted by θ in equation 2.1, were estimated based on their responses to the nanoquiz questions. While the IRT process does not directly equate number of correctly answered questions to a student's ability (θ), looking at the overall percentage of correctly answered questions on the nanoquizzes and seeing how that and the estimated θ values compare to each other can help in evaluating the validity of the estimation. Figure 4-1 plots exactly this data for all 29 nanoquizzes that occurred in the fall semester of 6.031 in 2019. This shows a high correlation between these two values both visually and numerically, as indicated by the r^2 value (0.91).

The estimated theta plotted against the student's overall grade in Figure 4-2 also is suggestive of some correlation though not as strong as the one depicted in Figure 4-1. Figure 4-2 is suggesting that while high final grades do not necessarily mean high ability, but it does seem to be in most cases that a high ability is indicative of a high overall grade. It is important to note that ability here is indicative of the latent ability in answering just nanoquiz question. The result of the plot is understandable, as a student's overall final grade is a combination of various parts of the course so a student may still do well on a course despite low performance on the nanoquizzes. However, it is interesting to note that students with relatively high ability ($\theta > 2.0$) earned some of the top grades in the class. The correlation between ability and final grade becomes less apparent for lower ability probably due to the fact that at that level of performance on the nanoquiz, there are many other components of the course through which a student can earn points.

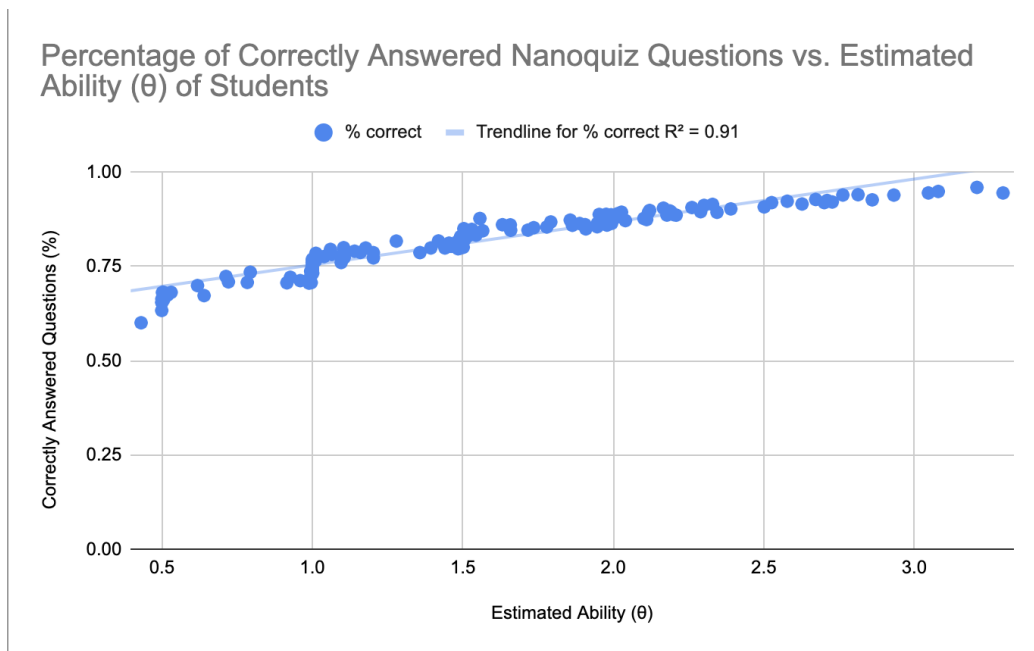


Figure 4-1: Plot of Percentage of Nanoquiz Questions Students Correctly Answered and Their Theta Values Estimated from Nanoquiz Responses (6.031 Fall Semester 2019)

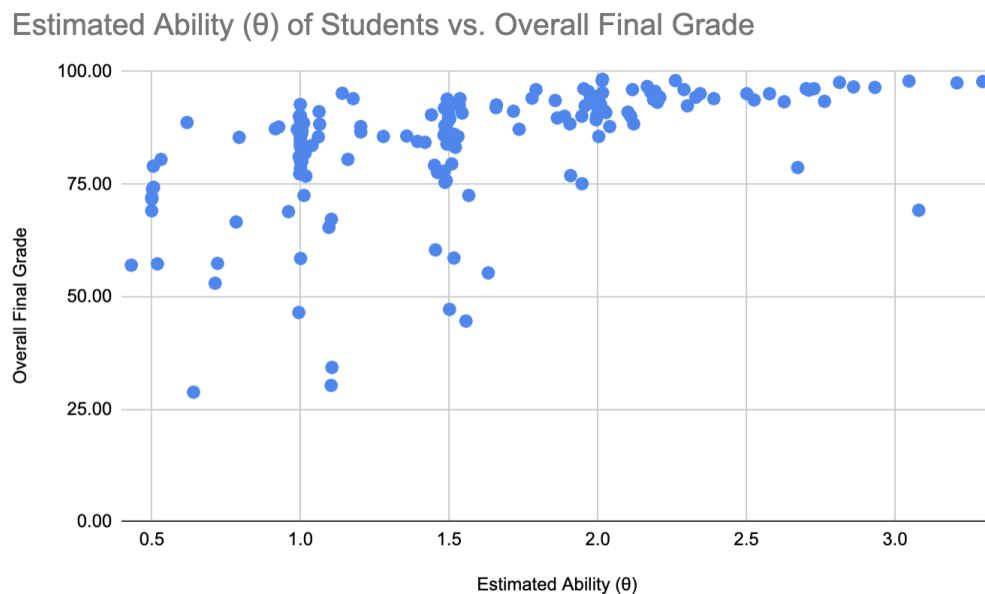


Figure 4-2: Plot of Final Grade for 6.031 and Their Theta Values Estimated from Nanoquiz Responses

4.2.2 Item Parameters (a and b) Estimation Results

Using the estimated item parameters of the nanoquiz questions, we can group questions by their estimated discriminability (a) and difficulty (b) values to see whether there are certain characteristics within each group. Figure 4-3 plots all the nanoquiz questions in their respective positions determined by their estimated a , b values. During the computation process, the boundaries of a and b were set to $[0.25, 2]$ and $[-2, 2]$, respectively. These values were chosen based on the ranges that these values realistically span [1]. Therefore, values will be clipped according to those boundaries.

The first group of questions to consider are the ones with relatively high a values. Most of the questions that fell into this range turned out to have high percentages of students answering those questions correctly, ranging from 81% to 100%. Many of the questions from nanoquizzes given during the first couple of classes which generally tend to cover easier and familiar topics that may have been taught in prerequisites of 6.031. Many of the questions in this category were usually asking the output of a simple code snippet. These questions are probably deemed to be able to discriminate students of different abilities well, as a lot of these questions were fairly easy to get right and therefore those students who answered incorrectly are more noticeable and their responses is indicated lack of understanding.

Items that have relatively low a values ranging from 0.25 to 0.35 tend to be questions that were complex. Questions in this range have a wider range of percentage of students who answered correctly. This could be due to the fact that most of the nanoquiz questions are written so a good percentage of the students will answer correctly. However, it is worthy to note that questions with low percentage of correctly answered students had low discriminability. These items covered more subtle concepts from the readings and often required students to apply definitions of concepts or to identify concepts given a concrete example. Some of the items were used as distractor choices, answer choices that were not correct at all. Additionally, some of these questions asked material that is very specific to the readings. The questions themselves do not necessarily suggest that questions with low discriminability are

Estimated Item Difficulty (b) vs Estimated Item Discriminability (a) of Nanoquiz Questions

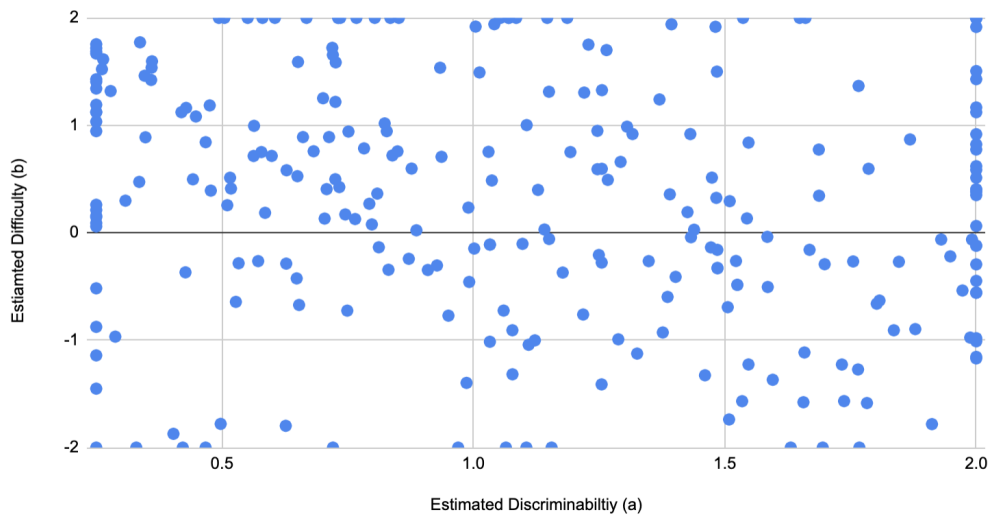


Figure 4-3: Plot of Estimated Item Difficulty (b) and Estimated Item Discriminability (a) of Nanoquiz Questions from 6.031 Fall Semester 2019

necessarily poor questions if one considers the purpose of the question. If the purpose of a question is, like the 6.031 nanoquizzes, to check whether a student has done the reading or not, questions in this category that ask specific things from the reading may be more appropriate than questions that may ask students to mentally execute simple lines of code.

Question items with higher b values also tended to be questions that asked about more simple topics whereas those with lower b values were often asking for answers specific to readings. This observation is not consistent with the description of the b value in IRT, and it is not clear why the observation is like so. In theory, decreasing b values are considered easier or more frequently endorsed by its respondents [2]. However, many questions with lower b values were requiring more complex skills, such as questions asking to correctly identify related concepts given an example of it.

Table 4.1: Minimum, Maximum and Median of the Estimated Item Parameters of the Nanoquiz Questions.

Measure	a (discriminability)	b (difficulty)
min	0.25	-2
max	2	2
median	1.1053	0.4037

4.3 Student-generated Questions

As outlined in section 4.1, the ability estimates from the student responses to nanoquiz can be used when estimating the item parameters (a and b) of the questions generated by the students from the question workflow. This is possible because the ability estimated by the IRT model is a measure of a latent quality of the students and is not item dependent. Therefore, these ability values can be used as given values when predicting the item parameters of the questions written by students [9].

There are some caveats, however, in this approach, unlike when we were predicting item parameters from the nanoquiz data. The setup of the workflow limits the number of total students answering the question to three and therefore, while in theory the group of respondents selected in predicting the item parameters should not matter, it is important to note this difference in the number of responses we have from the nanoquizzes which had more than 100 responses per item.

4.3.1 Categorization of Questions

From the more than 600 student-generated questions from the fall 2019 semester in 6.031, most of these questions can be categorized into one of the following: definition, recalling, identification and application. These categories were decided mainly based on the type of skills required by the question respondent. While there are certainly overlaps between the different types of questions and some questions often require multiple of these skills, questions are categorized by the main and more complex skill that is required to answer the question. For example, application questions almost always require correct recollection of facts, but the focus of those types of questions

is in one's ability to take these facts and apply it to new situations.

Definition types of questions are questions that test definitions of concepts that were outlined in the reading. Examples of such questions would often be in the form of fill in the blank where either the definition of a term or the term was mentioned and the corresponding definition or term had to be selected.

Questions under the recalling category require the answerer to recall facts that were mentioned in the reading. These types of questions can be answered if one can just remember the correct statements mentioned in the reading. These questions are often in the form of true or false questions where only the true statements have to be selected. Questions asking for definitions technically do fall under this category. However, the number of definition questions identified from the student-generated questions were significant enough to have its own separate category.

Identification questions are those that require the answerer to correctly identify concepts from some example that demonstrates such concepts. These questions were often paired with some snippet of code asking whether there were instances of concepts that were mentioned in the readings. A concrete example of this type of question related to 6.031 would ask whether certain segments of a method specification are part of the precondition or postcondition given a specification.

Application questions are more broad and varied in their form, but these are questions that are application of the concepts covered in the readings. These types of questions require both the writer and the respondent of the question to apply the ideas from the reading to a new environment. Some application questions from the student-generated questions often give fresh examples that demonstrate concepts from the class or requires the respondent to actually exercise the concepts discussed in the readings.

Although it was not a category of its own, questions that were not correctly formatted according to the Markdown syntax, as briefly mentioned in section 3.2.3, were also noted. Some ill-formatted questions that did not follow this syntax resulted in code blocks appearing as just plain text and not in a separate code block as it should. Other poorly formatted questions resulted in questions that were impossible

Estimated Item Difficulty (b) vs. Estimated Item Discriminability (a) of Student-generated Questions

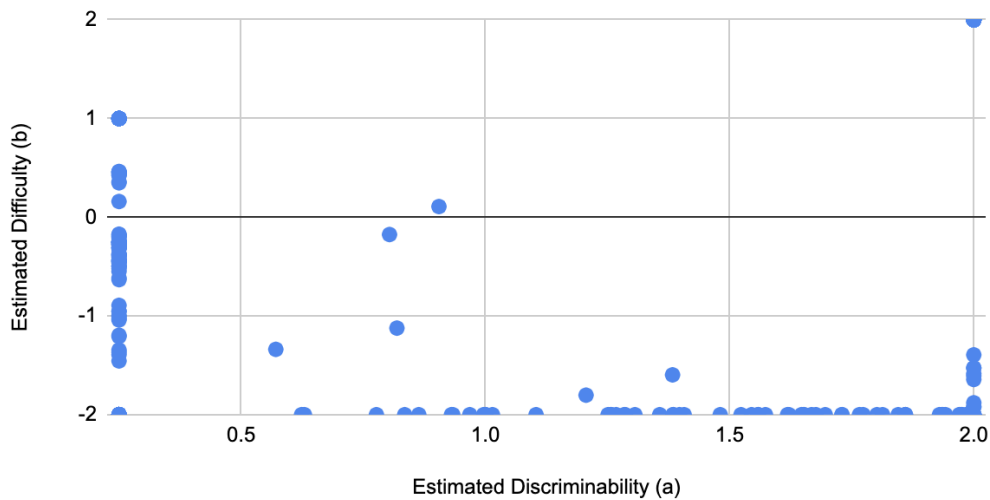


Figure 4-4: Plot of Estimated Item Difficulty (b) and Estimated Item Discriminability (a) of Student-generated Questions from 6.031 Fall Semester 2019

Table 4.2: Minimum, Maximum and Median of the Estimated Item Parameters of the Student-generated Questions.

Measure	a (discriminability)	b (difficulty)
min	0.25	-2
max	2	2
median	0.25	-2

to select options or answer explanations already marked before the answerer could attempt to answer the question.

4.3.2 IRT Estimation of Student-written Question Parameters

The estimated item parameters of the student-written questions were mostly around the lower boundaries of a and b in our implementation. Figure 4-4 and Table 4.2 both capture this aspect of the item parameters distribution.

Various aspects of a question can be measured and observed, and seeing the distribution of these measures across different ranges of a and b may provide some insight

about how the estimates of a and b may indicate some traits of a question. For example, we can see the distribution of question types across different ranges of a and b . For each question, there is also data from the playtest results, where a playtest refers to another student answering questions generated by their peers. As described in section 3.2.4, Questionable stores the following information from a playtest: number of attempts in answering a question, the result of each attempt, the comment left by the student. From this, it is possible to see different aspects of playtests for questions in various ranges of a and b .

Question Category Distribution Across a, b Values

Tables 4.3 and 4.4 both summarize the distribution of the types of questions in different ranges of a and b . Questions that require application of concepts generally tend to be the most common type of questions across the all ranges of a and most ranges of b . When looking at the various b ranges, application questions are clearly the most dominant in the following ranges of b : $b = -2$ and $-1 < b \leq 0$. The distribution among recalling, identification and application question types is more even in ranges $-2 < b \leq -1$ and $b = 2$. From this, it is hard to conclude any definitive statement about the relationship between the types of question and the range of b values. Definition type questions were mostly associated with lower b values, meaning less difficult questions. Identification questions are more prominent when the b values are positive. This is suggestive of the fact that these types of questions are more difficult for students to answer.

The distribution of a also suggests some relationship between a and definition and application questions. Application questions are also a popular question type regardless of the a values as well. As for definition questions, definition questions tend to have lower a values which are essentially questions with lower discriminability.

The question type distribution in regions of 1. high a and high b and 2. low a and low b suggests that low values in both difficulty and discriminability affect the types of submitted questions one would expect from students. The overall distribution is shown in Table 4.7. Again, low values in both item parameters have greater

Table 4.3: Question Categories Percentage By a ranges of Student-generated Questions

Category	$a = .25$	$.25 < a \leq .5$	$.5 < a \leq 1.5$	$1.5 < a < 2$	$a = 2$
Definition	19%	0%	0%	3%	4%
Recall	23%	0	31%	28%	26%
Identification	23%	0%	30.8%	0.287	32%
Application	53%	0%	39%	40%	38%

Table 4.4: Question Categories Percentage By b ranges of Student-generated Questions

Category	$b = -2$	$-2 < b \leq -1$	$-1 < b \leq 0$	$0 < b \leq 1$	$1 < b \leq 2$
Definition	2%	0%	3%	3%	2%
Recall	23%	38%	26%	17%	25%
Identification	22%	24%	11%	31%	36%
Application	53%	38%	61%	48%	36%

percentage of application questions whereas the distribution between identification and application is more spread out when both item parameter values are higher.

Table 4.5: Question Categories Count By a ranges of Student-generated Questions

Category	$a = .25$	$.25 < a \leq .5$	$.5 < a \leq 1.5$	$1.5 < a < 2$	$a = 2$
Definition	6	0	0	4	4
Recall	74	0	4	36	28
Identification	72	0	4	37	34
Application	168	0	5	52	40

Table 4.6: Question Categories Count By b ranges of Student-generated Questions

Category	$b = -2$	$-2 < b \leq -1$	$-1 < b \leq 0$	$0 < b \leq 1$	$1 < b \leq 2$
Definition	6	0	1	1	2
Recall	70	8	10	5	22
Identification	67	5	4	9	31
Application	159	8	23	14	31

Table 4.7: Question Categories Percentage By High a , High b and Low a , Low b of Student-generated Questions

Category	$a = .25, b = -2$	$a = 2, b = 2$
Definition	1.3%	1.5%
Recall	17.5%	16.4%
Identification	18.1%	23.1%
Application	41.3%	23.1%

Other Question Groups Distribution Across a, b Values

While not part of the four question categories mentioned in Section 4.3.1, there were other characteristics of questions that could be grouped together. These are questions that are incorrectly formatted, received borderline or minus grades or require a simple mental execution of a block of code. These types of questions were also separately noted because these factors can suggest the question quality, or lack thereof.

From the count of each group of questions as noted in Tables 4.8 and 4.9, qualities indicating a poor question are more often associated with lower a and b values. This seems to be more so for questions that involve mental code execution. In other words, these are questions that simply tested in one's ability to read code. Tables 4.10 and 4.11 also show the distribution across different ranges of a and b for each question group. Mental code execution questions are more likely to be associated with lower b values. While majority of the low grades (grades equivalent of borderline or minus) are associated with low a and b values, a good percentage of those low-grade questions are associated with high a and high b values. This seems to suggest that the criteria of grading is less dependent on the inherent properties, like difficulty and discriminability, than the a, b values measure. The distribution of these various question groups seems to suggest that simple questions or questions that are not

Table 4.8: Student-generated Question Count by Question Characteristics (Ill-formatted, low grades, simple code execution) Across a Values

Type	$a = .25$	$.25 < a \leq .5$	$.5 < a \leq 1.5$	$1.5 < a < 2$	$a = 2$
Mental Code Execution	66	0	2	15	14
Ill-formatted	21	0	3	1	7
Borderline/Minus	36	0	2	5	18

Table 4.9: Student-generated Question Count by Question Characteristics (Ill-formatted, low grades, simple code execution) Across b Values

Type	$b = -2$	$-2 < b \leq -1$	$-1 < b \leq 0$	$0 < b \leq 1$	$1 < b \leq 2$
Mental Code Execution	63	4	4	6	11
Ill-formatted	22	2	2	0	6
Borderline/Minus	36	5	5	2	13

formatted well are deemed to be less difficult or less discriminating against students of different abilities. These are generally qualities that we want to avoid when picking out good questions. These metrics can be used to filter out the bad questions, leaving us with a solid set of relatively good questions.

Table 4.10: Student-generated Question Distribution by Question Characteristics (Ill-formatted, low grades, simple code execution) Across a Values

Type	$a = .25$	$.25 < a \leq .5$	$.5 < a \leq 1.5$	$1.5 < a < 2$	$a = 2$
Mental Code Execution	68%	0%	2%	16%	14%
Ill-formatted	66%	0%	9%	3%	22%
Borderline/Minus	59%	0%	3%	8%	30%

Table 4.11: Student-generated Question Distribution by Question Characteristics (Ill-formatted, low grades, simple code execution) Across b Values

Type	$b = -2$	$-2 < b \leq -1$	$-1 < b \leq 0$	$0 < b \leq 1$	$1 < b \leq 2$
Mental Code Execution	72%	5%	5%	7%	13%
Ill-formatted	69%	6%	6%	0%	19%
Borderline/Minus	59%	8%	8%	3%	21%

Table 4.12: Grade Breakdown by Cause for Borderline and Minus Grades

Grade	Simple	Similar to Reading	Simple and Similar to Reading	Erroneous	Lack of Playtesting Effort	Irrelevant	Others
Borderline	36	10	1	5	2	2	5
Minus	14	21	4	4	3	4	4

4.3.3 Qualitative observations

Qualitative observations of the questions generated submitted by students suggest that students are often gravitate towards creating questions they have seen before. As mentioned in Section 4.3.2, application questions were most submitted by students. However, it was frequently the case that those questions were similar or identical to existing questions from the course readings. There were also instances where the student-generated questions were similar to the structure of the questions that the students were answering. This was an observation that was made in Peerwise as well [6]. In fact, one of the main reasons for student makeups to receive either a borderline or a minus grade is in either the question or the explanation, sometimes even both, being similar or same to the exercises in the class readings. The specific breakdown of the various reasons for a borderline or minus grade is shown in Table 4.12.

Table 4.13: Playtester Attempts Statistics

Minimum	Maximum	Average	Median
0.667	21	2.500	2

Table 4.14: Playtester Correct Answer Rate Statistics (measures the rate a student ends up answering a question correctly.)

Minimum	Maximum	Average	Median
0%	100%	92%	100%

4.4 Level of student engagement

Level of student engagement in the proposed workflow can be inferred by various metrics such as the effort put into playtesting and the grade they receive for their makeups. Efforts put into playtesting can be estimated by looking at whether they have tried to answer the questions by the number of attempts they make and the rate at which they try to attempt to get the question correct in the end. The average number of attempts a student makes when playtesting a question is around 2.5 where the median is 2 (Table 4.13). What is more significant perhaps is the rate at which they answer the question correctly in the end. Students on average end up answering the question correctly 92% of the times as shown in Table 4.14.

4.5 Changes in Grading Experience for Staff

Surveys from staff also reflect positive changes in grading makeups. Most of staff have expressed reduced time in grading and this can be attributed to mainly the change in 6.031's grading standards for makeups and also Questionable's grading user interface. First of all, the new grading standard now focuses on both the effort put into writing a question and the effort put into answering other questions and leaving comments. Determining effort in these aspects is now possible with the grading UI displaying the appropriate information in a user friendly way that helps the grader see whether a student has tried to attempt to get the questions correct. In addition, with other student playtesters of the question being able to comment on a student's

question, students are in a position to point out possible errors in the question. In fact, students have actually been leaving comments correcting or suggesting changes to the questions. Reading the comments has helped staff to corroborate the doubts and questions they have had themselves when evaluating the correctness of a question. In addition, staff have said that having student comments pointing out suggestions and potential errors in the student-written questions have minimized the task of pointing out mistakes and improvements for student. This resulted in saving time. Some staff, however, have mentioned that there is often the tendency to scrutinize a question to see if the question is correct and this still is a time-consuming process. In addition, not all comments always point out the mistakes, some even suggesting the wrong things. Therefore, some staff still feel the need to be responsible for correctly evaluating a the correctness of a question.

Chapter 5

Discussion

5.1 Conclusion

The proposed question generation workflow was introduced with the purpose to benefit both parties of teaching and learning. This workflow is supposed to target students and enhance their learning by exposing them to more questions from their peers and for them to build a better understanding of the class material by giving them the opportunity to engage with it by creating their own questions. The introduction of playtesting other student-written questions allows students to engage in the process of evaluating questions, and thus interacting with the class material in a new way. This also creates a new source of feedback to the original student authors of the question.

This peer feedback is not only useful to the student authors but also to the staff grading their work. Questionable's grading user interface presents all the relevant information about a student's makeup: the question they have written, the playtests done by other students and the student's playtest of other questions. Previously, staff were the only arbiters of a questions' correctness and much time was spent evaluating a question and leaving comments. However, with results of other student's playtests of the question the staff is grading, the staff can ascertain the appropriateness and quality of a question faster than before by reading the comments left by other playtesters and their attempt results. While this is a lot of information for the staff grader to digest, Questionable's UI presents these data in a user friendly way by

grouping relevant information together and displaying them effectively so this task of determining a student's grade and effort is not time-consuming.

In using IRT on the student responses to the nanoquizzes, it was possible to calculate estimates for the student abilities and the item parameters (difficulty and discriminability) for each answer choice for every question in all nanoquizzes. In particular, our implementation of the IRT computes values using only the responses actually made by the student. The estimated abilities seem to have high correlation with the overall percentage of correctly answered question items. There is also stronger correlation for students with higher overall final grade. Both of these observations suggest that the estimated ability values for each student from IRT are good estimates of student latent traits.

Nanoquiz questions of various a values seem to suggest varying level of question complexity, but this does not necessarily suggest that low a values are lower in question quality. Higher a values on nanoquiz questions were generally those that most students had gotten correct and were testing concepts that were fairly simple. These questions often consisted of material covered in the beginning of the course or those that simply require one to perform mental code execution. On the other hand, questions with lower a values were often asking about more subtle things or specific facts from the reading. On the other hand, the types of questions that fell into the various ranges of b were less consistent with the expected behavior. Most questions with high b values were often questions from the beginning of the course asking outputs of simple code blocks whereas questions with low b values asked specific things from the readings or required students to identify the correct concepts given examples of those.

Though the number of responses per student-generated question is low, looking at the distribution of the a , b values of the student-generated questions with respect to the type of skill the question requires to answer can provide some insight about the probable characteristics of questions that fall into various ranges of a and b . One interesting observation is that application types of questions, questions that require the answerer to apply existing knowledge to a new instance, were the most common

type of question. Application questions were mostly spread out across various a and b ranges. Simpler questions, such as questions that were testing definitions of terms, had mostly lower a and lower b values. This suggests that more simple questions of this sort are not discriminating and not difficult. Another interesting observation is the increasing portion of identification types question for higher a and b questions. According to the IRT metric, these questions would be more difficult to answer and better at discriminating students of different abilities.

A qualitative observation while grading is that most student-generated questions are not perfect and in order to filter for good questions, the initial step to take would be to look out for poor question using qualities that indicate such questions, like the grade of the makeup, the existence of ill-formatted questions. From this approach, we have noticed that questions that require simple mental code execution and ill-formatted were more prevalent with lower a and lower b values. While questions with borderline and minus grades were more observed in these ranges as well, it was not to the degree of mental code execution and ill-formatted questions.

These initial applications of IRT on the nanoquiz and student-generated questions provide some useful initial insights about how the various values of a and b can help in characterizing questions and determining the question qualities. However, there are limitations in some of the approach including the inconsistency with our observations of the question types and the values of b . Additionally, the low number of responses used to compute the item parameters may have an effect on the overall result, not allowing for computations to be run enough times correctly estimate the item parameters.

This new proposed workflow has helped out staff to become more efficient in grading by introducing students to be part of the provider of the feedback. Having comments and attempt histories made by students provide a greater insight into the quality of the questions which can be determined more quickly. This new workflow, paired with Questionable, has allowed staff to grade submissions faster and reduced the pressure of providing all useful feedback. This not only benefits staff but also students as well as since they are now able to receive feedback quicker than before

and from more people.

5.2 Future Work

While the initial execution of the proposed question generation workflow was successful, there are still various ways where the process can be improved. The first improvement is based on the observation that most of the student-generated questions still contain noticeable errors. While playtesters point out glaring issues of a question and sometimes the subtle correctness of the student-generated question and its explanation, there are instances where the wrong comment is given by others or issues critical to the correctness of the question is not addressed. In order to further solidify a student's understanding of the class material and to make use of the comments they have received, incorporating additional steps to the workflow to allow students to iterate on their question could not only help students practice their skills but also result in questions of better quality.

Another common reason for poor grades as outlined in Section 4.3.3 is poorly formatted questions. A solution to this would be to have stronger validation of the makeup form in Questionable that checks for whether the questions are formatted correctly, especially code blocks in the student-written questions. If the system is able to point out exactly where the formatting is off as the student is writing their question, this will push students to correct their formatting and reduce the instances where a student's experience in answering another student-generated question is not impacted by how the question is presented.

Many of the questions that received low grades were either due to its similarity to reading questions. A similar note of concern is the similarity between the student-generated questions and the questions they have submitted. Possible beginning steps to address this issue could be to have the system be able to detect similarity between the student-generated questions and the questions in the reading and those the student has playtested. If this is implemented with more accuracy, then staff can better determine quickly these instances and communicate to students about the

staff's expectations for more originality in the question. This communication can help students to be more cognizant about how they are creating their questions. Ideally, in the end, we would like a system where students are more moved to come up with more original questions of good quality.

Bibliography

- [1] Item response theory. <http://www.mailman.columbia.edu/research/population-health-methods/item-response-theory>. Accessed: 2020-04-30.
- [2] *The Basics of Item Response Theory*. The Art of Computer Programming. ERIC Publications, second edition, 2001.
- [3] Steven Bottemley and Paul Denny. A participatory learning approach to biochemistry using student authored and evaluated multiple-choice questions. *Biochemistry and Molecular Biology Education*, 39(5):352–61, September 2011.
- [4] Christine Chin and David E. Brown. Student-generated questions: A meaningful aspect of learning in science. *International Journal of Science Education*, 24(5):521–549, November 2010.
- [5] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. Peerwise: students sharing their multiple choice questions. In *Proceedings of the fourth international workshop on computing education research*, pages 51–58, 2008.
- [6] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. Quality of student contributed questions using peerwise. In *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95*, pages 55–63, 2009.
- [7] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. Codewrite: supporting student-driven practice of java. In *SIGCSE '11: Proceedings of the 42nd ACM technical symposium on Computer science*, pages 471–476, March 2011.
- [8] F. Dochy, M. Segers, and D. Sluijsmans. The use of self-, peer and co-assessment in higher education: A review. *Studies in Higher Education*, 24(3):331–350, 1999.
- [9] Brad Hanson. Irt parameter estimation using the em algorithm. 2000.
- [10] Judy Hardy, Simon P. Bates, Morag M. Casey, Kyle W. Galloway, Ross K. Galloway, Alison E. Kay, Peter Kirsop, and Heather A. McQueen. Student-generated content: Enhancing learning through sharing multiple-choice questions. *International Journal of Science Education*, 36(13):2180–2194, 2014.

- [11] Michael R. Harwell, Frank B. Baker, and Michael Zwarts. Item parameter estimation via marginal maximum likelihood and an em algorithm: A didactic. *Journal of Educational Statistics*, 13(3):243–271, 1988.
- [12] Henry L. Roediger III and Jeffrey D. Karpicke. Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological science*, 17(3):249–255, 2006.
- [13] Andrew Luxton-Reilly, Paul Denny, Beryl Plimmer, and Robert Sheehan. Activities, affordances and attitude: how student-generated questions assist learning. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pages 4–9, 2012.