

AutoER: A System for the Automatic Generation and Evaluation of UML Database Design Diagrams

by

Sarah Foss

BCIS, Okanagan College, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The College of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

July, 2022

© Sarah Foss, 2022

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

AUTOER: A SYSTEM FOR THE AUTOMATIC GENERATION AND
EVALUATION OF UML DATABASE DESIGN DIAGRAMS

submitted by SARAH FOSS in partial fulfilment of the requirements of the degree of Master of Science

Dr. Ramon Lawrence, I. K. Barber Faculty of Science
Supervisor

Dr. Patricia Lasserre, I. K. Barber Faculty of Science
Supervisory Committee Member

Dr. Bowen Hui, I. K. Barber Faculty of Science
Supervisory Committee Member

Dr. Stephen W. McNeil, I. K. Barber Faculty of Science
University Examiner

Abstract

Interactive question systems improve student engagement and provide opportunities for increased practice and skill mastery. Developing database design diagrams is a key skill for database courses, but providing evaluation feedback is time-consuming for instructors and accurate auto-grading is challenging due to the variability of student answers especially when labeling diagram components. This work presents a system for the automatic creation and real-time evaluation of database design questions using Unified Modeling Language (UML) diagrams. Students directly interact with the question text, and the system continuously generates a visual representation of their answer as well as provides immediate feedback at any time. By utilizing a web-based, customizable user interface, the system supports precise marking and the ability to practice variants of design questions to mastery. Classroom evaluations demonstrate high student satisfaction compared to traditional UML design questions and preference for using the software to improve their learning outcomes. Experiments also measured how submission limits, such as maximum submission limits and applying regression penalties, significantly improve student behavior.

Lay Summary

This thesis proposes and implements a new system, AutoER, for the creation and automatic evaluation of database design diagrams. This system allows students to build visual representations of diagrams by directly interacting with question text. Students can, at any time, request automated feedback to check their work. Instructors can choose to load the system with pre-written questions, or they can have the system auto-generate random questions. The ability to generate questions allows students to practice questions until mastery and also reduces academic dishonesty by generating unique questions for each student during examinations. Surveys from classroom evaluations revealed high student satisfaction and that students prefer to use the software over traditional methods. To limit over-reliance on the auto-grading feature, limitations on the number of student attempts per question were explored and measured in the classroom studies.

Preface

The study in this thesis was conducted with the approval of the UBC Okanagan Behavioural Research Ethics Board (BREB) under the certificate number H21-01600.

Parts of this thesis were published in the research-article titled *Automatic Generation and Marking of UML Database Design Diagrams* in the Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, Pages 626–632, in February 2022.

Table of Contents

Abstract	iii
Lay Summary	iv
Preface	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
Glossary	xii
Acknowledgements	xiv
Dedication	xv
Chapter 1: Introduction	1
1.1 Thesis Contributions	1
1.2 Thesis Organization	3
Chapter 2: Background	4
2.1 Database Design and Diagram Creation	4
2.1.1 Simplified Educational Database Design Tools	4
2.2 Automated Evaluation	7
2.2.1 Pattern Matching	7
2.2.2 Machine Learning	7
2.2.3 String Comparison	8
2.3 Generating Questions	10
2.4 Limitations on Submissions	10
2.5 Discussion	12

TABLE OF CONTENTS

Chapter 3: AutoER System Development and Design 13

- 3.1 System Overview 13
- 3.2 System Architecture 14
 - 3.2.1 Student Frontend 15
 - 3.2.2 Backend 15
 - 3.2.3 Database 17
 - 3.2.4 Reverse Proxy 17
- 3.3 Features 19
 - 3.3.1 Login 19
 - 3.3.2 Question Interface 19
 - 3.3.3 Submission Policies 21
 - 3.3.4 Student Mark Retrieval 23
- 3.4 AutoER Question Creation 23
 - 3.4.1 Question Markup 27
 - 3.4.2 Answer Format 28
 - 3.4.3 User Interface Template 30
 - 3.4.4 Automatic Question Evaluation 30
 - 3.4.5 Automatic Question Generation 31

Chapter 4: Case Studies: Undergraduate Database Course . . 34

- 4.1 Participant Recruitment 34
- 4.2 System Use Data Collection 34
 - 4.2.1 Assignments 36
 - 4.2.2 Midterm Exam 38
 - 4.2.3 Final Exam 38
- 4.3 Student Survey 39
- 4.4 Course Administrator Survey 40

Chapter 5: Results 42

- 5.1 System Evaluation 43
 - 5.1.1 System Use 43
 - 5.1.2 System Usability 44
 - 5.1.3 Free-form Survey Results 44
 - 5.1.4 Administrator Evaluation 46
- 5.2 Student Performance Evaluation 47
 - 5.2.1 Assignment Performance 47
 - 5.2.2 Midterm Generated Question Performance 48
 - 5.2.3 Final Exam Performance 48
 - 5.2.4 Summer 2021 Study 49
 - 5.2.5 Fall 2021 Study 52

TABLE OF CONTENTS

5.3 Discussion	61
Chapter 6: Conclusion	65
6.1 Limitations and Threats to Validity	67
6.2 Future Work	68
Bibliography	70
Appendix	76
Appendix A: Consent Forms	77
A.1 Administrator Consent	77
A.2 Student Consent	79
Appendix B: Surveys	81
B.1 Student Survey	81
B.2 Administrator Survey	83
Appendix C: Assignments	85
C.1 Assignment 1	85
C.1.1 Question 1 (10 marks)	86
C.1.2 Question 2 (10 marks)	87
C.2 Assignment 2	88
C.2.1 Question 1 (15 marks)	88
C.2.2 Question 2 (35 marks) - Project Deliverable	90

List of Tables

Table 3.1	AutoER Question Components	25
Table 5.1	Description of Common Metrics in AutoER Evaluations	42
Table 5.2	AutoER Usage Metrics	43
Table 5.3	Assignment Upload Metrics	43
Table 5.4	System Usability Scale survey results Categories - SD: Strongly Disagree, D: Somewhat Disagree, N: Neutral, A: Somewhat Agree, SA: Strongly Agree	45
Table 5.5	AutoER Performance Metrics	47
Table 5.6	Student Profile Categories	50
Table 5.7	Summer Student Profile Metrics	51
Table 5.8	Limiting Attempts Change	52
Table 5.9	Comparison of Profiles Across Studies	53
Table 5.10	User Count and Grades with One Submission	55
Table 5.11	Performance Comparison of Submission Policies	56
Table 5.12	Limiting Attempts Across Student Profiles	58
Table 5.13	Limiting Attempts: Student Preference and Performance	61

List of Figures

Figure 2.1	Database Design Text and UML Class Diagram	5
Figure 3.1	AutoER System Architecture	14
Figure 3.2	Administration Interface Question Creation	17
Figure 3.3	AutoER Design Diagram	18
Figure 3.4	AutoER Login Screen	19
Figure 3.5	AutoER User Interface	20
Figure 3.6	Adding a Key to an Attribute	20
Figure 3.7	Editing a Relationship between Entities	21
Figure 3.8	Example Student Feedback	22
Figure 3.9	Evaluation Details of a Student’s Submitted Answer	24
Figure 3.10	Question Base	27
Figure 3.11	Question Markup	27
Figure 3.12	Original Answer	29
Figure 3.13	String Representation of the Answer	29
Figure 3.14	Check Mark Button	30
Figure 3.15	Generated Question Example with Letter Placeholders	33
Figure 4.1	Assignment 1 Question 1	37
Figure 4.2	Final Exam Question and Answer	41
Figure 5.1	Student Performance Grouped By Total Submissions. Marker shows maximum quartile submissions cutoff.	49
Figure 5.2	Example Student Performance Profiles	50
Figure 5.3	Example Fall Student Performance Profiles	54
Figure 5.4	Midterm Exam Students by Category and Submission Policy	57
Figure 5.5	Final Exam Students by Category and Submission Policy	57
Figure 5.6	Midterm Exam Students by Category and Submission Policy	59

LIST OF FIGURES

Figure 5.7	Final Exam Students by Category and Submission Policy	60
Figure C.1	Assignment 1 Question 1	86
Figure C.2	Assignment 1 Question 2	87
Figure C.3	Assignment 2 Question 1	89
Figure C.4	Assignment 2 Question 2	91

Glossary

Application Programming Interface (API) A software intermediary with a set of definitions and protocols to allow applications to interact. 15–17, 30

Data Definition Language (DDL) The syntax for creating and modifying database objects. 90

Entity Relationship (ER) A database design model that describes how entities relate to each other. 3, 4, 6, 7, 9, 30, 44, 47, 65, 85, 90

Hypertext Markup Language (HTML) A markup language used to structure web documents. 14, 30

Internet Protocol (IP) The set of rules for routing and addressing packets of data. 35

JavaScript Object Notation (JSON) A text-based format for representing structured data. 15, 16

Learning Management System (LMS) A software application designed for the distribution, management, and automation of educational materials. 14, 26

Massive Online Open Course (MOOC) An open online course designed for large numbers of geographically dispersed students. 1

Representational state transfer (REST) An architectural style defining standards used across web-based applications. 15

Structured Query Language (SQL) A programming language used to manipulate and manage databases. 34, 37, 88, 90

System Usability Scale (SUS) A standardized survey tool used for measuring system usability. 39, 44, 61, 67

teaching assistant (TA) An individual that assists an instructor with instructional activities. 1, 23, 40, 48, 62, 66

Unified Modeling Language (UML) A modeling language used to express and design software systems. iii, 1, 4, 6–9, 12, 13, 23, 25, 34, 36, 48, 65–68, 85, 90

Uniform Resource Locator (URL) A web address. 15–17, 19, 25, 26, 32, 66

Acknowledgements

I would like to express my deepest gratitude to Dr. Ramon Lawrence. It has been a privilege to work with and learn from such a dedicated educator and researcher. His passion for teaching and commitment to his students' success is inspiring, and I hope to bring it with me and emulate it in my own academic career. Without his wealth of experience, guidance, and patience this work would not have been possible.

Special thanks to Tatiana Urazova. Her invaluable assistance and dedication helped ensure the success of this work. I would also like to acknowledge Dr. Youry Khmelevsky for his continued support of the project.

Thank you to my supervisory committee, Dr. Bowen Hui and Dr. Patricia Lasserre, for reviewing my work and helping me ensure that it is the highest quality possible.

This endeavor would not have been possible without the ongoing support of the faculty and administration at Okanagan College, and in particular, the support of my colleagues Ken Chidlow, Dr. Jim Nastos, Dr. Scott Fazakerley, and Alan Kennedy.

Finally, I could not have undertaken this journey without the support of my family. Thank you to my amazing husband, Jeramie, and my two incredible children, Gabrielle and Gavin. Their unending patience and unwavering support mean the world to me.

Dedication

For Anne and Clair Foss.

I'm so incredibly fortunate to have had your love and unconditional support in my life. I know you would be proud.

Chapter 1

Introduction

Classrooms can now take many different formats ranging from traditional on-campus lecture halls to online learning platforms offering Massive Online Open Courses (MOOCs). Many courses, regardless of the format, can have hundreds of students, creating time-consuming workloads for course administrators. These large class sizes can also present issues with precise marking as it is difficult for humans to grade a large quantity of assignments or exams consistently, particularly when the marking workload is divided across multiple instructors or teaching assistants (TAs).

Automated learning tools can help alleviate these issues by providing consistent, instant feedback to students. These tools can further benefit students by providing opportunities for increased practice with learned concepts and skill mastery. Several systems support the generation and auto-marking of questions [WHZ15, EP08, BE15] for practice and improving learning outcomes.

Database design and modeling are key concepts taught in database and software engineering courses. Design exercises are often challenging for students and additional practice helps learning. This work focuses on questions where the goal is to produce a design diagram for a database that captures the requirements discussed in a written text. Evaluating design diagrams is complex due to the lack of testing frameworks, diversity of correct answers, and interpreting the semantics of the diagram and its concepts. Prior work examined automatic evaluation of design diagrams using image processing [TWS06], syntactic, structural, and semantic analysis [BAK19], and machine learning [BMM20]. These systems provide evaluations similar to human markers for instructor developed questions.

1.1 Thesis Contributions

The contribution of this thesis is a question generation and evaluation system for Unified Modeling Language (UML) database design diagrams and its evaluation in a database course. The system, called AutoER, supports both instructor-created and automatically generated questions with

instructor configurable marking. Automatically generating question variants allows for students to practice questions until mastery and is beneficial for online exams where the potential for student collaboration and academic dishonesty exists. AutoER provides flexibility for instructors to change all aspects of its behavior including the user interface, marking and feedback, question generation, and operation as a stand-alone system or integrated with a learning management system.

A unique aspect of the user interface is that the student builds the design diagram by interacting with the question text directly and requesting the system add components to the diagram rather than drawing them manually. This improves the speed of diagram construction and helps beginners learn fundamental design challenges. Consistent naming also enables precise marking and matching with correct solutions.

The system was evaluated in two offerings of an undergraduate database course at the Okanagan campus of the University of British Columbia. The research questions were:

- **RQ1:** Do students prefer to use the AutoER system compared to paper and other design software?
- **RQ2:** Is the AutoER system efficient and usable for students?
- **RQ3:** Does automatic grading increase student performance on database design assignments?
- **RQ4:** Is random generation of questions effective for use in exam evaluations?
- **RQ5:** Does limiting submissions to the system impact student submission behaviour in a positive way?
- **RQ6:** Which of the submission policies limiting student submissions is most effective?
- **RQ7:** What submission policy is preferred by students?

To address these questions, students were given an opportunity to use the system for assignments, examinations, and additional practice at their own discretion. Extensive data was collected regarding the system usage and student performance. The students were surveyed after several weeks of usage to gain insight into the perceived usability and effectiveness of the system.

1.2 Thesis Organization

This thesis examines the development and evaluation of an interactive pedagogical tool for the generation and evaluation of Entity Relationship (ER) diagrams. Chapter 2 presents a background on the database design problem and a discussion of existing systems for diagram creation and automatic evaluation. The AutoER system architecture and system features are described in Chapter 3. Chapters 4 and 5 detail the experimental setup and results using the system in an undergraduate database course. Finally, the conclusion of the thesis and potential future work are discussed in Chapter 6.

Chapter 2

Background

2.1 Database Design and Diagram Creation

A fundamental learning objective in an undergraduate database course is the ability to design a database for a particular problem domain. Database design is a challenging problem for students as they must learn the particular diagram syntax and be able to translate the written text into the appropriate diagram elements.

Database diagrams are constructed using either ER notation [Che76] or UML [Gro17]. UML class diagrams represent database entities as classes, relationships as associations, and attributes may be associated with entities and relationships. To manually construct a database design problem, an instructor would describe a database schema using plain language and would also construct one or more solution diagrams. A set of grading criteria would be defined to standardize the evaluation of answers. Student's diagrams would then manually graded by visual comparison between the student's answers and the solution diagrams. Evaluation and providing feedback is time-consuming as students may not use the syntax consistently, misinterpret the question, use inconsistent naming and labels, and not understand fundamental modeling constraints.

Figure 2.1 contains a database design question text and the answer as a UML class diagram.

2.1.1 Simplified Educational Database Design Tools

Given a particular database design question, an instructor allows a student to produce the diagram on paper or using UML database design software. Surveys of modeling software for instruction [ALS19, AL17] indicate preference for easy-to-use systems to allow students to focus on learning the concepts rather than the software. Other important aspects in preferred systems are that they are perceived to give good feedback, are simple to install, and that they should be low cost or free to use. Commercial products used by experts are often costly, large programs that contain more features

2.1. DATABASE DESIGN AND DIAGRAM CREATION

There are multiple hospitals in the medical system. A hospital is identified by its name and has a location.

A doctor is identified by their medical number and has a name. Each hospital has a single doctor as a manager, and a doctor may manage only one hospital.

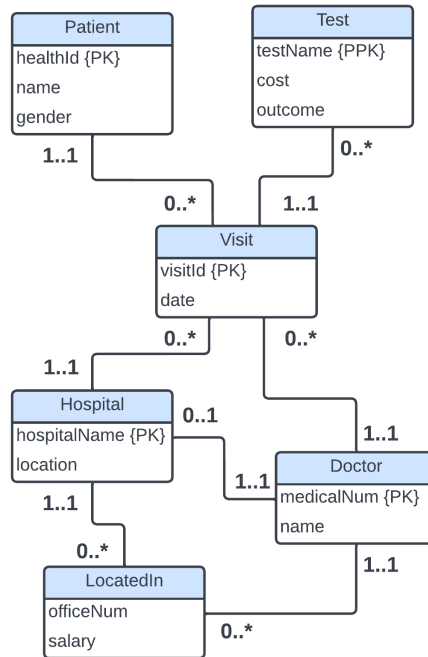
Doctors are located in hospitals. A doctor may be located in more than one hospital. A doctor located at a hospital has an office number and a salary paid by that hospital.

A patient is identified by their health id and also has a name and gender.

A patient visits a doctor at a particular hospital. Each visit is identified by its own id and also has a visit date.

At a visit zero or more tests are run each with a cost and an outcome. Each test is identified for a particular visit by name.

(a) Question Text



(b) Example UML Class Diagram

Figure 2.1: Database Design Text and UML Class Diagram

2.1. DATABASE DESIGN AND DIAGRAM CREATION

and complexity than required for educational use.

Several low-cost educational systems have been developed to support the construction and evaluation of database designs in ER or UML class diagrams.

QuickUML [AV03] was developed in response to existing systems that provide too much UML notation to beginner students, over-complicating the learning process. This system simplifies beginner level learning by only providing basic UML functions to allow students to begin to apply newly learned concepts without being slowed down with more advanced aspects of domain modeling. This tool was intentionally kept small and lightweight to allow for easy installation on a number of systems. Although this system is lightweight and easy to use, feedback to students is not a system feature.

The tool MinimUML [TPE05] takes a similar approach and only provides a subset of UML notation with a focus on high-level abstract design. The developers of this tool noted that some existing simple tools were lacking some of the desirable features that more complex systems have and created a system that supports features like printing and undo/redo. The system also supports annotations that allows instructors to provide feedback, but automated feedback is not supported.

DiagrammER [GM20] is another simplified educational tool, however, it is a web-based application which allows users to avoid installation of the tool altogether. This system allows beginner computer science students to create and interact with ER diagrams by dragging and dropping elements in the diagram creation area. A novel feature of the system allows instructors to upload example questions with the corresponding correct ER diagram for students to retrieve and view. Unfortunately, automated feedback is not a feature that is implemented in the system; however, it is planned future work.

EERTutor [Zak04] is a tutoring system for ER design that provides interactive diagram construction and tutorials to help students during diagram creation. When an error is made, the toolbar and editing area are disabled and a dialogue with common-sense feedback is presented to the student. There are multiple levels of feedback available. The first level of feedback simply informs the student whether the answer is correct. Students can then choose the option for a “hint” or a “detailed hint” for more detailed information about the error and guidance to improve their answer. If the student is still unable to answer the question, they can request to view the solution diagram. These prompts help the students identify and correct errors when they may not have enough experience or knowledge to do so on their own. This level of in-depth feedback is beneficial for formative assessments, but

may not be appropriate for summative evaluation.

2.2 Automated Evaluation

Providing automatic feedback and evaluation for modeling and design tasks has received increased attention in recent research. Similar to grading and evaluation systems for questions on programming [Lee21, MNS⁺20, EP08, MSR21], providing real-time feedback to students allows for formative learning, increased engagement, and consistency and efficiency in evaluation. Automatic evaluation of design diagrams must match student solutions with one or more instructor solutions while handling issues with naming, structural, and semantic differences.

There are several approaches for automatic grading and feedback for diagrams.

2.2.1 Pattern Matching

Marking of student drawn ER diagrams and other diagram types using image processing techniques and recognizing patterns that match with answers was discussed in [TWS06, STW13]. This approach does not require systems that enforce precise diagrams, meaning diagrams where the names of diagram components are pre-defined. To perform the evaluation, the student diagram and the answer diagram are compared for similarity by identifying components that convey specific meaning in that domain and comparing them to the matching components in the solution. Similarity between entities is measured by the weighted sum of their label identifiers and attributes and similarity between relationships is measured by the weighted sum of the labels of objects they relate and their cardinalities. The result is used to generate a grade and feedback for the student. Thresholds must be defined to determine whether two components are sufficiently similar to be considered a correct match. Challenges of using pattern matching with imprecise student diagrams include accurately grading images due to missing or extra information, and inconsistent terminology including synonyms, abbreviations and misspelling, and determining optimal thresholds for similarity.

2.2.2 Machine Learning

In [SvdPSC19], machine learning was used to grade UML class diagrams. In their approach, multiple classification models and a regression model

were trained with extracted features from student answer files from a single assignment along with associated grades from expert markers. Experiments were done using both a 10-point scale (with grades ranging from 1-10) and a 5-point scale. Results showed that accuracy was not sufficient with a top accuracy of 42% with the 10-point scale and 46% with the 5-point scale.

Machine learning automatic assessment in [BMM20] produced grading by letter grade. This research compared five common machine learning models trained with student answers along with grades produced from a heuristic string parsing grading algorithm. The output was then compared to manual instructor-graded solutions. Results showed that both Logistic Regression and Naive Bayes models were accurate within a letter grade to the baseline. However, only letter grades were provided, and there was no feedback for improvement to students. Another big drawback was that the dataset had to be extensively manually cleaned to correct misspelling, missing syntax, and duplicate entries in order to be used with the simple heuristic algorithm.

2.2.3 String Comparison

Most commonly, student and instructor diagrams are compared using a string representation of the diagram contents.

UMLGrader [Has11], which is based on UMLint [HR11], a tool used to identify defects in UML diagrams, is a web-based application that evaluates UML class diagrams developed using IBM Rational Rose. Each diagram in Rational Rose has an associated model file that describes the diagram in text. Using the model files from a solution diagram and a student diagram, the text is compared using string comparison. For the comparison, spaces and underscores are stripped and capitalization is ignored. Entity names require exact matches, and attributes and associations require partial matches with the associated names from the solution. Because students manually enter component names for the diagrams into the system, they are required to name their diagram precisely otherwise it may be marked as incorrect. The system does flag common errors like misspelling and invalid multiplicities using UMLint, however, the tight constraints of the system along with manual name entry pose some challenges to the auto-evaluation system.

[SBP⁺10] presents a similar UML database design tool allowing students to create diagrams with string comparison auto-marking. This tool records all of the parameters of an answer into a file and compares it directly with the solution file. In this system, specific attribute names, denoted with

2.2. AUTOMATED EVALUATION

parenthesis, must appear in the problem description to be valid for use in the problem. This addresses one of the challenges present in UMLGrader, but also unfortunately eliminates the exercise of the identification of attributes for the student. The constraint of manually entering exact names of entity classes is still present in this system.

Another system using string comparison for automatic evaluation is MonstER park [Sch20]. This tool uses gamification to encourage student learning by interactively developing a diagram corresponding with a story. As the game is played, both a visual diagram display for the student and an underlying structure comprising of JavaScript arrays that represent the elements of the ER diagram are built. The user inputs are normalized by removing all non-alphabetic characters and then converted to lowercase text. The auto-evaluation is performed by comparing the string contents of the answer arrays to the strings in solution array. Although exact matching is required for this system as well, there are a number of acceptable matches in each solution array to allow for slightly more flexibility in the student's answer.

A system was developed [BAK19] that attempts to address the problem of exact naming in auto-grading software. This tool use syntactic, structural, and semantic matching of string representations of UML diagrams to allow students to enter a wide range of answers that may be evaluated as correct by the system. The system uses the Levenshtein distance [L⁺66] to determine a syntactic match for component names. Names are also semantically matched using a combination of three algorithms to determine a similarity score. The classes are then subject to a structural match by comparing component names and the matched classes are then compared and graded. To evaluate the auto-grading system, 20 diagrams were graded manually by an instructor and then assessed again with the tool. The automated grades were within 14% of the instructor's grades.

The auto-grading accuracy was improved by further customized grading criteria [BAK20]. In this study, experiments were conducted on two different groups to measure the auto-grading effectiveness of the system compared to manual evaluation. There were two sets of evaluations done. First, the diagrams were graded without the marking customization. Then, they were evaluated again with the instructor's custom grading criteria. In the first evaluation, the first group had results that had an average difference of 6.5% to the instructor's grading and the other group's results had an average difference of 24.6%. After adding the custom marking configuration, the difference in the first group was reduced to 4.8% and the difference in the second group was reduced to 13%. This addition to the system in-

creased accuracy for the auto-grader to be within 5 to 15% of instructor grades. However, although this seems to be relatively high accuracy, any grade deviation from the instructor’s grade is undesirable and because of the variation of inputs accepted into the system, higher marking accuracy may be difficult to achieve with this approach.

2.3 Generating Questions

A barrier to providing a large quantity of database design questions to allow students to practice concepts to mastery is that instructor-generated questions and marking criteria are time-consuming to produce. The capability to generate question variants based on parameters is valuable as shown by domain independent systems such as PrairieLearn [WHZ15] and Web-CAT [EP08]. Generated question variants support repetitive student learning and are useful in evaluation situations to reduce student collaboration and copying [CWZ18]. To the author’s knowledge, there has been no prior research done in the specific area of database design question generation.

2.4 Limitations on Submissions

Auto-grading systems provide students the opportunity to utilize feedback to help guide their answers while practicing new concepts. However, if a system does not have a submission policy to limit attempts, students may develop an over-reliance on the feedback which may negatively impact their learning. Most automated assessment systems support either unlimited submissions or a maximum number of attempts per question as the default. Systems may also enforce a time limitation between submissions [EP08]. Another technique is to penalize mistakes where a student’s submission receives a lower mark than the previous submission, called a regression [BZHH21]. Limiting the number of submissions is widely used to counteract poor behaviors. Students are less likely to guess and more likely to spend more time thinking about a problem with limits applied. The goal of submission policies is to encourage independent student learning and reflection and reduce reliance on feedback and hints provided by the system. Automated assessment systems have the potential to improve positive student metrics [LSTA21] such as performance, engagement, learning, and retention and decrease procrastination and failures. Prior results have discussed how students may use feedback to change their solution without thinking about the problem and stop working on a question once the mark is “good enough”

2.4. LIMITATIONS ON SUBMISSIONS

[HBTN21]. Systems vary the amount of feedback and hints provided to reduce reliance on the system. Submission policies are also a key component in insuring positive outcomes.

Edwards [Edw04] argues for encouraging students to move from trial-and-error strategies to reflection in action. Prior research by Malmi et al. [MKKN05] demonstrates how introducing limits on attempts versus unlimited submissions encouraged students to spend more time on each submission and their first attempts had higher performance. Students often use trial-and-error near the end of an assessment to improve their grade but not their learning. In [KKM06], students were classified into five categories based on their number of submissions and final grade: *passers* - few attempts/low grades, *ordinaries* - few attempts/medium grades, *talented* - few attempts/high grades, *ambitious* - high attempts/high grades, and *iterators* - high attempts. This classification did not consider the time between student submissions or any grade difference between submissions. The classification goal was to identify user profiles and trial-and-error behaviour.

Detecting and penalizing regressions in student submissions that resulted in lower marks was explored in [BZHH21]. In this study, student performance on automatically graded programming assignments with an imposed regression penalty in a software engineering course were compared with the same assignments from the previous year without the regression penalty. Introducing the regression penalty was shown to dramatically reduce the number of submissions by 50%, decrease regressions by 90%, and increase the time between submissions. Regression penalties encouraged students to spend more time thinking about their answers before submission. However, it should be noted that some students indicated a perceived compromise to their grades as well as increased stress-levels related to the penalties. Further to that, the overall average grade fell by 10% and analysis on the assignment submissions revealed a reduced number of submissions after students achieved a high, but not perfect mark. A number of students reported that stopped submitting on the assignment after achieving 95% out of concern of lowering their grade.

The key challenge is identifying the appropriate submission policy that leads to desirable outcomes. Unlimited attempts and flexibility are common for formative assessments [MBF⁺18] where the focus is on student practice and learning. Appropriate feedback allows students to improve their understanding [MBF⁺18], increases engagement [MGF⁺20], and reduces bias and variation in grading. For formative assessments, the submission strategy should discourage poor behaviors [Auv15, MDP20] such as the trial-and-error strategy and procrastination. Students may be more likely to “game

the system” [Pie13] compared to learning the material with an automated system compared to instructor grading. Multiple attempts improve student performance in summative assessments [FGF21] but determining the number of attempts may be a harder issue to address. A survey of assessment systems [IAKS10] explicitly mentions limiting the number of submissions and the amount of feedback as considerations for these systems, but there is a lack of recommendations and best practices.

2.5 Discussion

As noted above, there are several tools for pedagogical diagram creation and evaluation, however, no existing system contains all of the features that is presented in this work. Like a number of the tools described above, AutoER is a simplified database diagram creation tool that is easy to use, requires no installation, and is free for student use. It also provides automated feedback to students that can be requested at any time. This work is different from the aforementioned tools in two main regards:

1. Students do not manually enter component names. Instead, they interact with the question text directly to identify the diagram elements. This ensures that the system’s string comparison approach to auto-evaluation of diagrams is accurate and precise. Since the class names and attributes are predefined in the question, there is no need for syntactic or semantic matching which means that student answers can be directly compared with solutions without ambiguities and the marking will match the instructor’s rubric exactly.
2. The AutoER system also is capable of automatically generating UML diagram design questions and the associated solution diagrams. To the author’s knowledge, this is a novel feature that is not present in any other database design learning system.

Chapter 3

AutoER System Development and Design

3.1 System Overview

The AutoER system supports students learning database design diagrams by providing real-time visualization and feedback during question answering. It is a simplified diagram creation tool that targets users first learning the design concepts and process of extracting model elements from written text. Students can answer instructor-created or randomly generated questions by interacting with the question text to develop the design (see Figure 3.5). Unlike other existing UML design software, the student selects the modeling construct to add (entity, attribute, relationship) from the question text but does not directly draw the construct on the diagram itself. The diagram is drawn based on the student answer dynamically. Because the answer is built with predefined keywords, the system is able to accurately and precisely evaluate the student's answer by string comparison to instructor solutions, avoiding the need for syntactic and semantic matching and other types of normalization on element names. In addition to this, there are also no other current systems that allow for the auto-generation of UML diagram design questions and solutions, which makes direct comparisons between AutoER and other existing systems difficult.

The key features are:

- Ability to create a question type that represents a particular question structure
- Support for uploading user interface code file and marking code file for a question type
- Creation of generated questions with solutions using uploaded code
- Storing, evaluating, retrieving student questions and answers
- Ability to optionally enforce attempt limiting submission policies

3.2. SYSTEM ARCHITECTURE

From the instructor perspective, the AutoER system is completely configurable including the user interface presented to students, the evaluation and feedback, and the parameters and approach for question generation. The system consists of a backend web server and database for generating, evaluating, and storing questions and answers. The JavaScript/Hypertext Markup Language (HTML) user interface allows students to interact with questions on any browser or Learning Management System (LMS), avoiding the hassle of program installation on their systems.

3.2 System Architecture

The server is implemented using Django, React, Traefik, and PostgreSQL as a stand-alone system deployed using Docker Compose. Porting the server functionality to use other question systems such as [WHZ15, EP08] is future work.

The system is comprised of four Docker containers: Student Frontend, Backend, Database, and Reverse Proxy as shown in Figure 3.1. These containers and their functionality are discussed below.

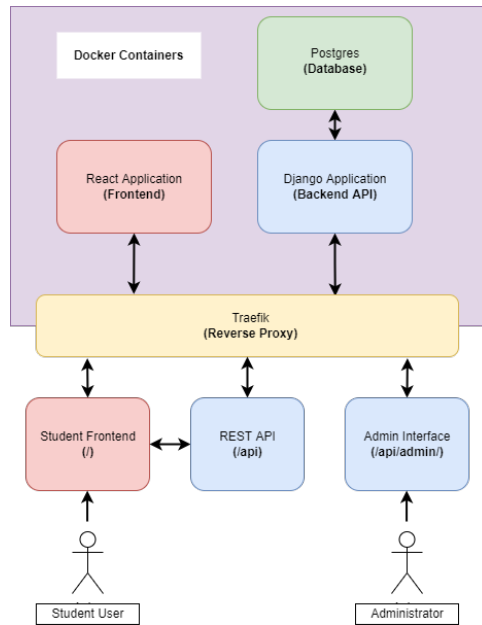


Figure 3.1: AutoER System Architecture

3.2.1 Student Frontend

The Student Frontend container hosts a React JS website that primarily handles and renders the question interface. After a user is authenticated, a React page retrieves and displays the instructor-defined template and question associated with the Uniform Resource Locator (URL). It then attempts to retrieve the last answer submitted by the authenticated user and associated auto-grading information for the question. If no answer is found, it will send an empty object to the displayed page denoting that it is the user's first attempt.

When a student submits an answer for evaluation, it sends the student's answer information to the Django component for evaluation and then awaits the marking information to send to the page for display.

This container also allows for course administrator authenticated access to pages that offer in-depth details about specific student answers and allows administrators to access a student's last answers for each question.

3.2.2 Backend

The Backend container hosts a Django application to handle the server-side processing for the AutoER system. The server's duties include the structuring of data, interfacing with the PostgreSQL database, communicating with the Frontend container, executing the auto-marking code, executing the question generation code, and providing the interface for administrators to manage questions and users.

REST API

This application allows for communication with the Frontend container through its Representational state transfer (REST) Application Programming Interface (API) built with the Django REST Framework. The frontend site will initiate a communication with an authenticated GET request at the server's /api endpoint and the server will respond by transferring the requested data as a JavaScript Object Notation (JSON) object.

Auto-Marking Answers

When the server is requested to evaluate a student's answer, it will retrieve the question, the custom Python auto-marking code associated with the question, and all the possible answers from the database. It will then compile and execute the code using the question and answers as arguments

and return a JSON object containing the evaluation results. The resulting grade information is then stored in the database and the results are sent as part of the API's response JSON object.

Question Generation

The process for question generation is slightly more complex. When the server receives a request for a generated question type, it will first attempt to retrieve the generated question associated with the user and question type denoted by the URL from the database. If it does not exist, then the question generation code will be run. The steps are as follows:

- Generate a random seed for the question to ensure uniqueness.
- Retrieve the associated generation type information from the database. This contains the generated code description (i.e., midterm exam question, final exam question, practice question, etc.), a reference to the associated question template, the maximum grade that the generated code should allow, and the custom Python question generation code.
- Execute the question generation code, passing the seed as an argument.
- Store the resulting generated question to the database using the seed as a title.
- Store the resulting answer to the database with a reference to the associated question.
- Add an associative entry to the database containing a reference to the question type, a reference to the generated question, and the current user.
- The generated question is sent as part of the API's request fulfillment.

Administration Interface

Currently, course administrators use Django's integrated administrator's interface to upload and update questions, question templates, and update all other question related information. Figure 3.2 shows an example of question creation in the administration portal. They also use the administration interface to create and update student user profiles. Future work includes building a custom course administrator's front-end to increase administrators' ease of use.

3.2. SYSTEM ARCHITECTURE

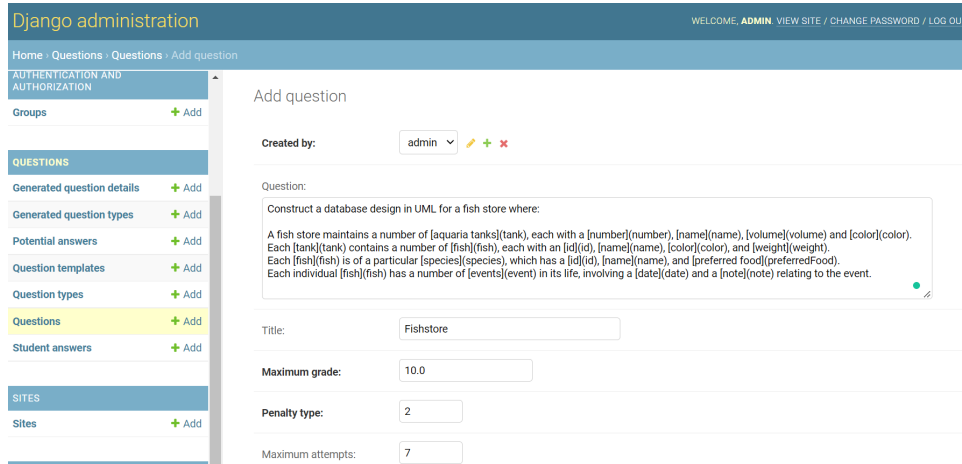


Figure 3.2: Administration Interface Question Creation

3.2.3 Database

The Database container hosts a PostgreSQL database server. As shown in Figure 3.1 this container only interfaces with the Backend container because Django effectively manages all the incoming and outgoing data using Object Relational Mapping.

Figure 3.3 shows the design diagram of the AutoER system.

3.2.4 Reverse Proxy

The Reverse Proxy container hosts a Traefik reverse proxy server. This server is the entry point into the AutoER system's multi-container environment and is the only accessible container from outside Docker. This container will map external URLs to internal Docker container services and route inbound traffic to the correct container.

The URL <https://autoed.ok.ubc.ca/> is the student endpoint and any requests made to that address will be routed to the Student Frontend container. The API and administration interface endpoint is the URL <https://autoed.ok.ubc.ca/api/> and the reverse proxy server will route those respective requests to the Backend container.

3.2. SYSTEM ARCHITECTURE

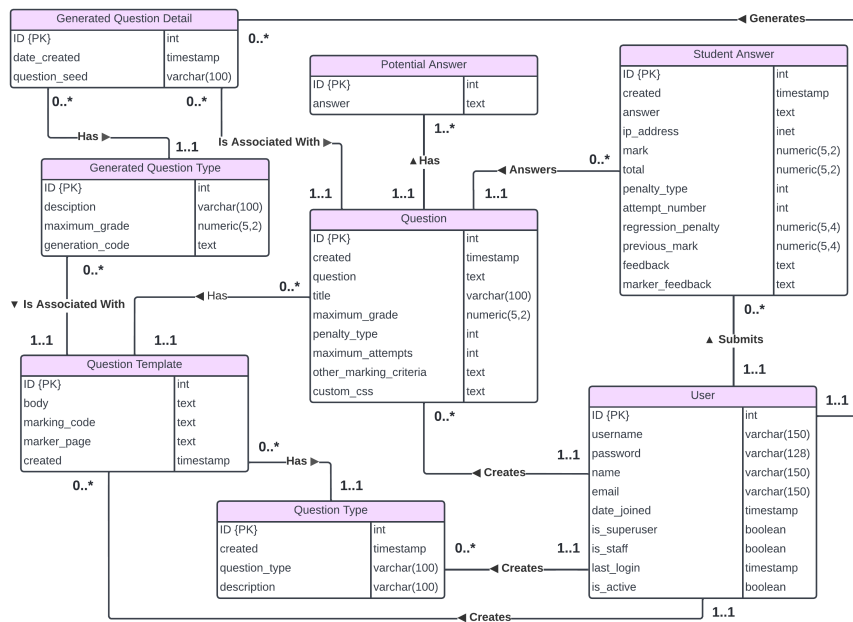


Figure 3.3: AutoER Design Diagram

3.3 Features

3.3.1 Login

A student accesses an AutoER question through a question URL that is provided by the instructor as part of an assignment or quiz. Each question has its own unique URL. Before they have access to any question, they must enter their preassigned username and password into the secure login screen, as shown in Figure 3.4. Any failed attempts will redirect the user back to the login screen with a notification of the incomplete sign-in. After a successful login, an authentication token is stored in local storage to allow the student a persistent logged in state.

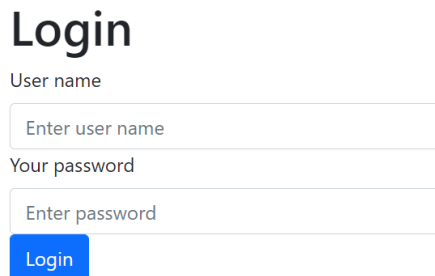


Figure 3.4: AutoER Login Screen

3.3.2 Question Interface

Once logged in for the first time, the student will see the question text and an empty diagram. The question text contains markup that allows students to select particular words and phrases to add to the diagram. Context-sensitive pop-up menus allow for adding entities and attributes. Figure 3.5 shows the user interface and a student answering the question. As the student builds the diagram, it is dynamically updated in real time. Diagrams are drawn using the open source *nomnoml* library¹.

There are two side menus that allow for editing entities and relationships between entities. The manage entities menu enables users to remove entities and attributes and add or edit key properties on attributes. Figure 3.6 shows a user using the manage entities menu to indicate a particular attribute is a key for an entity.

¹<https://nomnoml.com/>

3.3. FEATURES

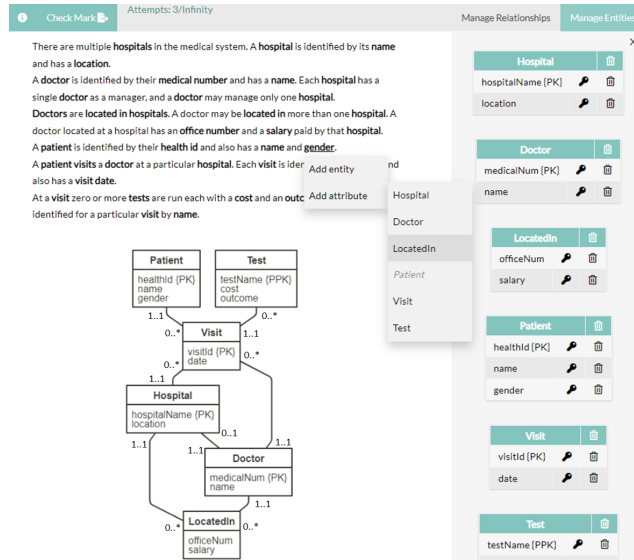


Figure 3.5: AutoER User Interface

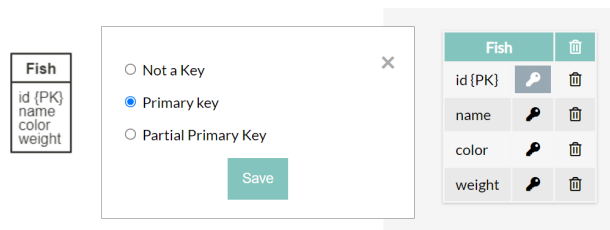


Figure 3.6: Adding a Key to an Attribute

3.3. FEATURES

Relationships can be added and edited between entities using the manage relationship menu. Relationships can be between two distinct entities or can be self-referential. The following cardinalities can be optionally added to relationships: zero or one (0..1), one (1..1), zero to many (0..*), and one to many (1..*). An example of a user editing a relationship between entities can be seen in Figure 3.7.

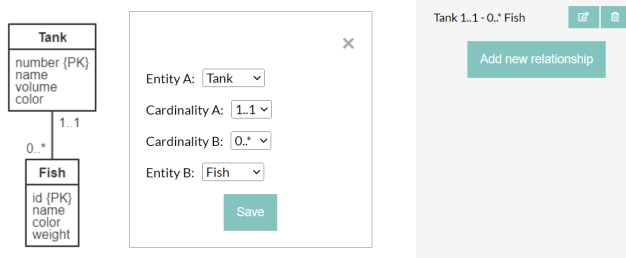


Figure 3.7: Editing a Relationship between Entities

When a student wants to request feedback, they click on the Check Mark button in the top left-hand corner as shown in Figure 3.5. This submits a text representation of the diagram for evaluation to the server and feedback is returned immediately to the student. The granularity of the feedback, marking strategy, and number of times a student can submit are controllable by the instructor. An example of feedback provided to students is shown in Figure 3.8. This high-level feedback provides only grading information at the level of diagram components (entities, relationships, etc.). It is suitable for exam situations, whereas feedback on specific errors may be more appropriate for formative assessments.

3.3.3 Submission Policies

A recent improvement on the AutoER system is the addition of new submission policies to limit the number of submissions per student. The attempts can be limited by either restricting the number of attempts or with a grade regression penalty. This is chosen by the instructor during question creation.

When the restricted number of attempts policy is enforced, the system will increment the attempt number after each submission made by the student. When the maximum number of attempts has been reached, the system will disable editing of the diagram and show the student the mark of the most recent attempt. The maximum number of attempts is customizable

3.3. FEATURES

Entity name marks:	1.2 / 1.2
Entity attribute marks:	0.5 / 0.6
Entity primary key marks:	0.8 / 1.2
Weak entity key marks:	0.5 / 1.0
Extra entities:	3.0 / 4.0
Total entity marks:	3.0 / 4.0
Relationship entity marks:	2.5 / 3.5
Relationship cardinalities marks:	2.5 / 3.5
Extra relationships:	-0.25
Total relationship marks:	4.75 / 7.0
Total marks:	7.75 / 11.0
Total scaled marks:	7.05 / 10.0

Figure 3.8: Example Student Feedback

by the instructor.

The regression penalty submission policy, based on the regression penalty introduced in [BZHH21], applies a permanent penalty to the student’s grade when a student submits an answer that is graded lower than the previous attempt. Where P is the previous higher mark and C is current lower mark, the regression penalty R is calculated as:

$$R = \frac{(P - C)}{2} \tag{3.1}$$

For example, if a student’s mark decreased from 94 to 90 from one submission to the next, then the regression penalty is 2 marks. The highest mark they would be able to receive would be limited to 98. Any regression penalty accrued is permanent and cumulative. The regression penalty could also be modified with different values, which may be investigated in future work. Although the regression penalty policy allows for an unlimited number of attempts by the student, it is intended to limit trial and error submissions.

During question creation, the instructor can choose from the following possible schemes:

- Unlimited attempts
- Restricted number of attempts
- Regression penalty
- Allow the student to choose regression penalty or limited attempts

3.4. AUTOER QUESTION CREATION

- Randomly assign a limited number of attempts set by the instructor or a regression penalty

In previous iterations of the system only the unlimited attempts submission policy was offered. As discussed in the results chapter, during the initial evaluation of the system, a high average number of submissions by students was noted. These policies were developed to limit undesirable submission behaviour.

3.3.4 Student Mark Retrieval

Instructors and TAs are able to access a detailed breakdown of the evaluation of a student's submitted answer. Figure 3.9 shows an example of the course administrator's view of a student's answer. This page shows the student's final mark, the text representation of their answer, the visual representation of their answer, the feedback the student received, and a Marker Feedback section with more in-depth information about the grading not available to the student.

The Marker Feedback section lists any missed or incorrect entities, attributes, relationships, and cardinalities and shows the correct diagram constructs from the answer. This allows any course administrator to verify the auto evaluation done by the AutoER system and answer student inquiries about their automated grade on a question.

3.4 AutoER Question Creation

Instructors define questions in the Django administration portal. Course administrators can also upload custom evaluation code, question generation code, and user interfaces. Table 3.1 summarizes the different customizable components involved in question creation and customization and the overall design of the database is shown in Figure 3.3.

Two types of database design questions can be created in the system: instructor designed questions and generated questions.

Instructor Defined Questions

Existing UML questions that have been written by instructors are convertible for use in the AutoER system. The steps for developing instructor-created database design questions are:

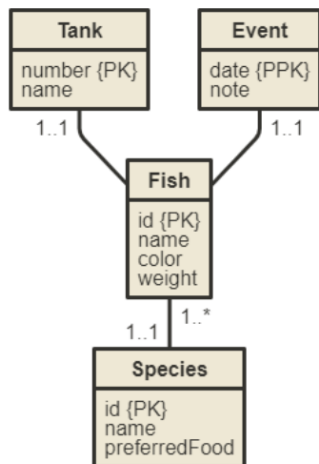
3.4. AUTOER QUESTION CREATION

Mark:

Answer:

```
[Tank|number {PK};name]
[Fish|id {PK};name;color;weight]
[Species|id {PK};name;preferredFood]
[Event|date {PPK};note]
[Tank]1..1 - 0..*[Fish]
[Fish]1..* - 1..1[Species]
[Event]1..1 - 0..*[Fish]
```

Diagram:



Feedback:

```
Entity name marks: 0.8/0.8
Entity attribute marks: 0.3/0.4
Entity primary key marks: 0.8/0.8
Weak entity key marks: 0.5/0.5
Extra entities: 0.0
Total entity marks: 2.4/2.5
Relationship entity marks: 1.5/1.5
Relationship cardinalities marks: 1.0/1.5
Extra relationships: 0.0
Total relationship marks: 2.5/3.0
Total marks: 4.9/5.5
Total scaled marks: 8.91/10.0
```

Marker Feedback:

```
Incorrect attributes on the Tank entity
Student: name, number
Correct: color, name, number, volume

Incorrect cardinalities on the ['Event', 'Fish'] relationship
Student: [Event] 1..1 - 0..* [Fish]
Correct: [Event] 0..* - 1..1 [Fish]
```

Figure 3.9: Evaluation Details of a Student's Submitted Answer

3.4. AUTOER QUESTION CREATION

Question Component	Description
Question Type	The type of question in the system. Currently, the system only accepts UML database diagram design type questions, but support for other types of questions is future work.
Question Template	The user interface for questions. Each question template also holds the marking code and the marker's feedback page and is associated with a question type.
Question	The marked-up text of a question, maximum grade, penalty type, and other question details. Each question is assigned to an existing question template.
Answer	An answer to an associated question. There may be multiple answers for one question.
Generated Question Type	A description for a question that may be randomly generated by a user and the code for generation. Every generated question type created has a URL for a user to visit to generate a random question. Each generated question type is associated with an existing question template.
Generated Question Detail	Details regarding a question that has been automatically generated including the generating user, date, and a reference to the generated question.

Table 3.1: AutoER Question Components

- The UML database design question texts are marked up with AutoER notation.
- The answers to the questions are converted to the required string representations.
- If the proper template does not already exist, a question type and template is created in the AutoER system using the administration interface. Each template contains the customizable code for: the question interface page, the auto-marking code, and the marker feedback

3.4. AUTOER QUESTION CREATION

page.

- The database design question is created in the system referencing the associated question template. Each question contains the marked up question text and specific marking criteria.
- The answers are created in the system referencing the associated question. Each answer contains the converted string representations of the answer diagram.
- The question is then made available to students using the question URL https://autoed.ok.ubc.ca/questions/{question_id} allowing access with any web browser or integration into a LMS. The system supports multiple question URLs that students can visit anytime.

Automatically Generated Questions

In addition to instructor-designed questions, AutoER is capable of automatically generating question variants. They allow students to generate any number of questions to allow them to practice until mastery. Generated questions are also useful for online exams to combat academic dishonesty as each student will be assigned a unique variant.

The steps for developing automatically generated database design questions are:

- If the proper template does not already exist, a question type and template is created in the AutoER system using the administration interface. Each template contains the customizable code for: the question interface page, the auto-marking code, and the marker feedback page.
- A generated question type is created in the system referencing the associated question template. Each generated question type contains the description of the question, the maximum grade for the generated question and the customizable code for the generation of a database design question.
- A question is made available to students using the question URL https://autoed.ok.ubc.ca/generatedQuestion/{gen_ques_id} allowing access with any web browser or integration into a LMS.

3.4.1 Question Markup

In many cases, instructors already have a selection of questions that they would like to assign their students. These pre-defined questions can be used with AutoER with the proper formatting.

A question can be converted for use by the system by marking up keywords that should be selectable by the student with square brackets. The set of marked-up keywords must include all of the entities and attributes that should be present in the final answer. It should be noted that every marked-up keyword can be added by the user as a possible entity and a possible attribute to any and all entities so other keywords that are not part of the answer can be marked up to increase the difficulty of the question. If there are keywords that are not appropriate for the graph (i.e., long keywords, multiple word keyword, etc.) alternate words can be substituted by including them in parentheses after the keyword entry. Figure 3.10 shows a pre-defined question before conversion and Figure 3.11 shows the question after conversion for use with the AutoER system.

```
Construct a database design in UML for a fish store where:  
  
A fish store maintains a number of aquaria tanks,  
each with a number, name, volume and color.  
Each tank contains a number of fish,  
each with an id, name, color, and weight.  
Each fish is of a particular species,  
which has a id, name, and preferred food.  
Each individual fish has a number of events in its life,  
involving a date and a note relating to the event.
```

Figure 3.10: Question Base

```
Construct a database design in UML for a fish store where:  
  
A fish store maintains a number of [aquaria tanks](tank),  
each with a [number](number), [name](name), [volume](volume) and [color](color).  
Each [tank](tank) contains a number of [fish](fish),  
each with an [id](id), [name](name), [color](color), and [weight](weight).  
Each [fish](fish) is of a particular [species](species),  
which has a [id](id), [name](name), and [preferred food](preferredFood).  
Each individual [fish](fish) has a number of [events](event) in its life,  
involving a [date](date) and a [note](note) relating to the event.
```

Figure 3.11: Question Markup

The question is loaded into the system by creating a question in the administrator's portal. The instructor is able to upload a formatted ques-

tion, set the total marks for the question (which will scale automatically), set the type of limiting penalty to be used and the maximum attempts if necessary. The question needs to reference the associated question template during question creation.

3.4.2 Answer Format

Every question entered into the AutoER system must have at least one associated answer. Multiple correct answers are also accepted by the system. Answers, which are typically in diagram form, must be converted into a string representation with the following rules:

- An entity is represented by enclosing the name in square brackets.
- Attributes are associated with an entity by adding a pipe character (‘|’) after the entity name and listing the attributes delimited by semicolons.
- Keys are denoted after the attribute name with either a {PK} or {PPK}, representing a primary key or partial primary key respectively.
- Define relationships between entities by specifying the two entities in square brackets and separating them with a dash.
- Cardinality constraints can be added to a relationship by including them on the appropriate side of the dash. Acceptable cardinalities are 0..1, 1..1, 0..*, and 1..*.
- Flexible cardinality constraints are also supported for questions with multiple correct answers. AutoER’s flexible cardinality constraints are 0@1, identifying that either 0..1 or 1..1 is correct and * identifying that either 0..* or 1..* is correct.

Figure 3.12 shows an example of an answer before conversion into the AutoER answer format. Figure 3.13 shows the AutoER string representation of that answer. The answers are loaded into the system by creating answers with reference to the associated question in the administrator’s portal. Future work includes the creation of an instructor front-end that would allow a question-and-answer string to be built dynamically by interacting with context menus, similar to the student question interface.

3.4. AUTOER QUESTION CREATION

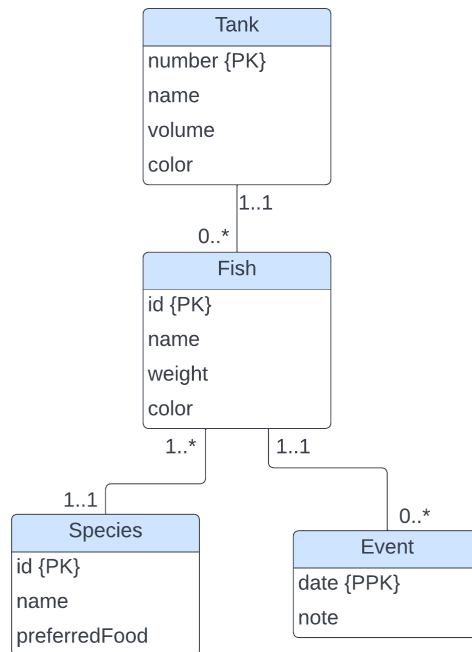


Figure 3.12: Original Answer

```
[Tank|number {PK};name;volume;color]
[Fish|id {PK};name;weight;color]
[Species|id {PK};name;preferredFood]
[Event|date {PPK};note]
[Tank]1..1 - 0..*[Fish]
[Fish]1..* - 1..1[Species]
[Fish]1..1 - 0..*[Event]
```

Figure 3.13: String Representation of the Answer

3.4.3 User Interface Template

The customizable user interface template of AutoER is implemented in a single JavaScript/HTML file. It is formatted as a single file to allow for simple uploading to the server.

When the file is first loaded by a browser, the file receives a marked-up question, any previous grade information and answer strings for the current user from the backend API via the React application. It will then parse the received marked-up question and reformat it with HTML for AutoER functionality. The question will be rendered with event-listeners on the keywords to add the interactivity with the question. If it is not the student's first attempt on the question, the last submission information will be displayed, and the previous attempt's diagram will be drawn to the HTML canvas element. Otherwise, the text "Your ER Diagram is empty" will be displayed on the canvas and it will display that it is the user's first attempt.

The system uses the `nomnoml` library to draw the visualizations to the canvas HTML based on a string representation of the answer. As the user interacts with the text of the question and menus, the program builds the answer string dynamically. After any updates to the string, the graph visualization is redrawn.

When the user clicks on the check mark button, as shown in Figure 3.14, the page submits the user's information to the Backend API via the React application for auto-evaluation and awaits the response. When the response is received, it displays the mark and feedback for the student. If there are remaining attempts, the template file allows the student to continue editing their diagram.



Figure 3.14: Check Mark Button

3.4.4 Automatic Question Evaluation

When a student requests feedback on a design question, the text representation of their answer is sent to the server for evaluation. The evaluation process performs the following steps:

- Matches entities in the student answer by name with instructor solution(s).
- For each entity, matches attributes within the entity.

3.4. AUTOER QUESTION CREATION

- Detects relationships in student answer matching instructor solution using names of entities and cardinality constraints (0..1, 1..1, 0..*, 1..*). Flexible cardinality constraints are supported for questions with multiple correct answers.
- Verifies cardinality constraints match solution.
- If more than one instructor solution is available, student receives highest mark when comparing with each solution.

The automatic question evaluation using the text representation of the answer has similar characteristics as other systems [BAK20, Has11]. Marks are assigned based on a student answer matching the solution’s diagram elements (entities, relationships, attributes), and the instructor assigns weights to each of the diagram elements. The default marking scheme assigns equal weight for all elements of the same type (e.g., all entities have same mark value), but instructors are free to modify marking criteria. Assigning marks equally makes it more efficient for instructors creating marking criteria. The system automatically scales the mark to a target question mark. An instructor can configure the weights of diagram elements, and even modify the Python code for individual questions for specific use cases.

The matching process has very high precision as the names and labels in the diagram are extracted from the question markup rather than free-form entry by students. This avoids issues with naming conflicts and matching ambiguity that reduce accuracy in other systems. The disadvantage is that the question markup provides students hints on the key information in the question. Instructors control the question markup and are encouraged to add markup for words that are not part of the solution to increase the difficulty of the question. Given a particular markup word phrase, the student must perform the key design activity of determining how to model as an entity, attribute, or relationship.

3.4.5 Automatic Question Generation

AutoER is capable of automatically generating question variants given parameters such as the number of entities, attributes, and relationships. Python code generates the question text and question answer using these parameters. Randomization allows different diagrams to be generated for each student.

Question generation has the steps:

3.4. AUTOER QUESTION CREATION

- Generate a random number of entities in range [min,max] given by instructor.
- Generate a random number of weak entities.
- Generate relationships of particular association cardinalities (0..1, 1..1, 0..*, 1..*). Rules limit the number of relationships between entities and number of recursive relationships.
- Names of entities, attributes, and relationships used in the answer and question text are randomly generated. Figure 3.15 shows an example of an automatically generated question with letter placeholders for entity and attribute names. Other placeholders, like “gibberish” words, may be used instead.
- Vary question text by randomizing phrase descriptions. For example, a 0..* relationship between entities A and B may be described as “A may be related to many B” or “A has multiple relationships with B” or other similar phrases.
- Question text statements are re-ordered to mimic real questions. For example, relationship information may appear near the associated entities or later in the description.

The server supports generated questions by allowing the upload of a code file for question generation. When the student accesses the question URL, the generation code is run to create a random question specific for that user that is stored in the database. Then the student interacts with the question as if created by an instructor.

3.4. AUTOER QUESTION CREATION

Check Mark
Attempts: 2/7
Manage Relationships
Manage Entities

A has key **a1** and has attributes **a2**, **a3**, **a4**.
B has key **b1** and has attribute **b2**.
C is identified by **c1** has fields **c2**, **c3**, **c4**.
D has key **d1** has fields **d2**, **d3**, **d4**. **D** may be related to many **B**, and **B** has at most one connection with **D**.
E exists dependent on **B** and has identifying attribute **e1**. **E** has fields **e2**, **e3**. **E** must be related to exactly one **A**, and **A** must be related to exactly one **E**.
G exists dependent on **D** and has identifying attribute **g1**. **G** has field **g2**.
A must be related to exactly one **B**, and **B** must be related to exactly one **A**.
C may be related to many **D**, and **D** has at most one connection with **C**.
D may be related to many **G**, and **G** is connected with one **D**.
C has multiple relationships with **G**, and **G** is connected with one **C**.

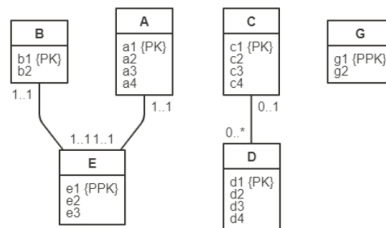


Figure 3.15: Generated Question Example with Letter Placeholders

Chapter 4

Case Studies: Undergraduate Database Course

The AutoER system was evaluated in two sessions of the same undergraduate database course, COSC 304 Introduction to Databases, at the University of British Columbia Okanagan.

4.1 Participant Recruitment

The first session was in the summer 2021 term with 44 students enrolled, and the second session was in the fall 2021 term with 180 students enrolled. After receiving consent from the instructor (see Appendix A.1), the students were given an introduction to the system at the beginning of the course by the author and were given the choice to participate in the study. The students that were interested in participating were asked to read and sign the consent form (see Appendix A.2). Although this study took place in a Computer Science course at the University of British Columbia Okanagan, no course administrators or participants of the study were involved in the development of the system or in the investigation of the studies. Specifically, the author or supervisor were not involved as instructors in the course.

4.2 System Use Data Collection

The database design module is about 20% of the COSC 304 course content and is taught after students have learned Structured Query Language (SQL) and the relational model. Students have limited prior experience with UML, which they are exposed to briefly in prior programming courses.

Data Collected

The following is a list of all the data collected from each submission to the AutoER system:

4.2. SYSTEM USE DATA COLLECTION

- User ID
- Question ID
- Submission number
- Student answer
- Automatically assessed grade
- Generated feedback for the student
- Detailed feedback for course administrators
- Date and time submitted
- Source Internet Protocol (IP) address
- Penalty type (second study only)
- Regression penalty (second study only)

Marking Scheme Used

Although the marking scheme is customizable, each diagram question in the study was marked with the following default marking scheme:

- 0.2 marks for each correct entity name
- 0.1 marks for each correct attribute list for an entity
- 0.2 marks for identifying each primary key
- 0.5 marks for identifying each weak entity
- -0.25 marks for each extra entity in diagram
- 0.5 marks for identifying relationship between two entities
- 0.25 marks for identifying each correct cardinality on relationships
- -0.25 marks for each extra relationship in diagram

If the total marks did not match the instructor defined mark of the question, the question's total calculated score was automatically scaled to match.

A question in AutoER may also be subjected to a submission policy of either a regression penalty, which applies a permanent penalty to a student's marks when a submission with grade decrease occurs, or a maximum attempt limitation. Like the marking scheme, the number of maximum submissions is customizable, but the default maximum number of submission used in the system for the policy was seven. The maximum attempt limitation of seven submissions was chosen as a baseline as it was a reasonable number designed to limit guessing but fewer attempts could be investigated in future comparison and study.

4.2.1 Assignments

In the course, there are two assignments involving design questions, a question on an online midterm exam, and a question on an online final exam. Students participating in the study were given the option of using the AutoER system for any of these components, or they could continue to answer the questions on paper or using the UML software previously used in the course. The assignments consisted of human generated questions consistent with prior years. The instructions and questions were presented on GitHub with a link to AutoER and also links to other software that students may choose to use to complete the questions.

The fall session study added some additional experimentation from the initial summer session study. In the second study, there were additional submission policies enforced in the system that were not present in the first study. This allowed for further analysis on the submission behaviour noted in the summer.

The first assignment was done in pairs and consists of two database design questions. Each question was marked out of 10. Figure 4.1 shows the first question of the first assignment and the associated solution diagram. See Appendix C for the detailed assignment questions and solutions used in the study.

The second assignment (Appendix C.2) was the first deliverable of the course project developing a database-driven, online store and was done in groups of up to four students. The second assignment has an advanced database design question involving multiple weak entities for the first question, and a large design question (11 entities, 13 relationships) for the second question. This assignment also requires using data types for attributes

4.2. SYSTEM USE DATA COLLECTION

Construct a database design in UML for an Online Game store described below.

A **developer** where each **developer** is identified by an **id** and has a **name**.

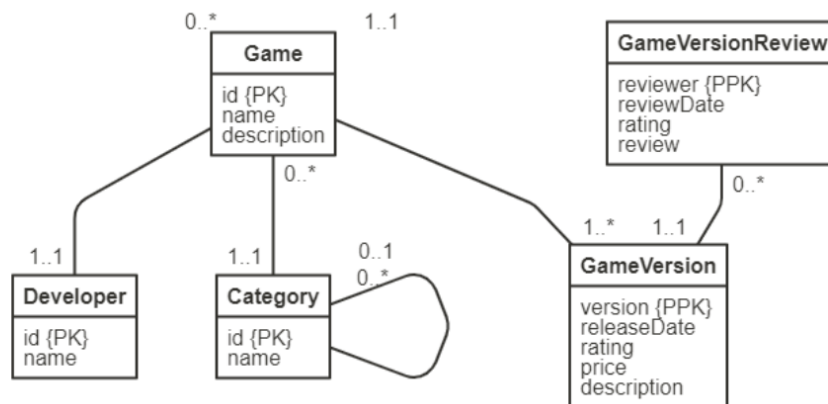
A **category** where each **category** has an **id**, a **name**, and may have a parent **category**.

A **game** storing each **game** that is identified by a field called **id** and other attributes include **name** and **description**. A **game** is created by one **developer**. A **developer** may publish multiple **games**. A **game** has a **category**.

A **game version** stores each version of the game. A **game version** is associated with exactly one **game**. Use a **version** field to identify between **versions** of the same **game**. Each **game version** has a **release date**, a **rating**, a **price**, and a **description**.

A **game version review** stores **ratings** for each game version. Each **instance** applies to a single **game version**, and different **reviews** are identified by **reviewer** attribute (which is name of reviewer). There is also a **review date**, **rating**, and **review**.

(a) Question 1



(b) Question 1 Answer

Figure 4.1: Assignment 1 Question 1

and mapping to the relational model, which are not supported by AutoER. Students using AutoER must manually convert their diagrams into SQL CREATE TABLE statements. The AutoER portion of the questions were worth 10 marks and 21 marks respectively.

The participants in the fall study and summer study both had unlimited attempts with no penalties on the assignment questions.

4.2.2 Midterm Exam

The midterm exam question used the random question generation from the AutoER system, but students were still able to answer the question however they wished. In COSC 304, the midterm exam is 90 minutes, open book, and students may use any software and course resources. Overall, 50% of the exam marks are related to database design with the other half on database programming. The randomly generated question was worth 20% of the exam.

In the summer study, participants who chose to use the system had unlimited attempts with no penalties on the midterm question. Student participants in the fall study were randomly assigned a submission policy of either a maximum of seven attempts or regression penalty. However, due to an error with the exam deployment on the learning management system, the split was not even. 29% of participants were assigned the maximum attempts limitation and 71% were assigned the regression penalty. As this was the first usage of the submission policies in the system, it is difficult to gauge the impact of the uneven split.

Before the summer exam, a practice midterm with an automatically generated question was available with the same format. For the fall exam, a practice midterm with three automatically generated questions were available: a practice question with a regression penalty, a practice question with a maximum of seven attempts limitation, and a practice question where the students could choose the submission policy.

4.2.3 Final Exam

In both sessions, the final exam was open book, and the final exam question in the AutoER system was an instructor defined question used in previous offerings of the course. Figure 4.2 shows the final exam question used in both studies. The question was presented on the online exam web page with a link to AutoER as well as instructions to upload an image of the diagram.

In the summer, the final exam was 3 hours and the database design question was worth 10% of the exam. Participants who chose to use the AutoER system had unlimited attempts with no penalties. A practice exam was available with the same format.

The final exam was 2.5 hours in the fall, and the database design question was worth 12% of the overall exam. For this exam, participants were able to choose the submission policy: they could either have a maximum of seven attempts or they could have unlimited attempts with a regression penalty. They were also able to choose their submission policy on the practice exam.

4.3 Student Survey

Students were asked to complete a survey (see Appendix B.1) after working with the system for several weeks. The survey contained the questions from the System Usability Scale (SUS) [Bro96], an established and widely-used questionnaire that measures ease of use in systems. It is comprised of 10 questions with Likert scale answers. To calculate a SUS score, each answer is converted into numerical data with the following scoring:

- Strongly Disagree: 0 points
- Disagree: 1 point
- Neutral: 2 points
- Agree: 3 points
- Strongly Agree: 4 points

Each score is adjusted by subtracting 1 from each odd-numbered score and subtracting each even-numbered score from 5. The sum of the adjusted scores is computed and multiplied by 2.5 to get the standard SUS score which will be in the range of 0-100. Equation 4.1 shows a more concise way of calculating the SUS score.

$$SUS = 2.5((SUS01 + SUS03 + SUS05 + SUS07 + SUS09) - 5) + (25 - (SUS02 + SUS04 + SUS06 + SUS08 + SUS10)) \quad (4.1)$$

Results of empirical evaluation of the System Usability Scale [BKM08] indicate that systems that score between 60 - 70 on the scale can be considered marginally acceptable and systems that score above 70 are considered acceptable, with better systems scoring in the high 70s and above.

The student survey also contained a multiple-choice question about their use of the system and five open-ended questions asking about why they used

4.4. COURSE ADMINISTRATOR SURVEY

the software, positives and negatives with the software, and general comments. The survey was made available after the midterm and was completed by 27 students out of 44 students (61%) in the summer offering and was completed by 50 out of 180 (28%) in the fall session.

4.4 Course Administrator Survey

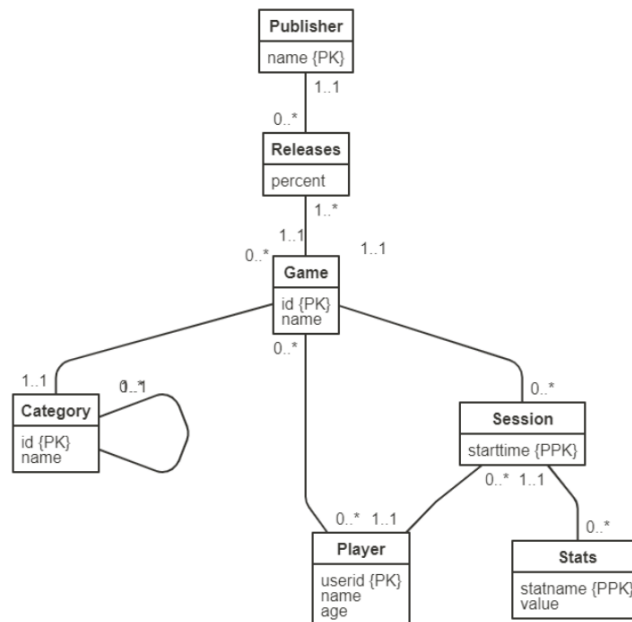
Instructors and TAs were asked to complete a survey (see Appendix B.2) after the end of the course. This survey was designed to gauge the effectiveness of the system from the course administrator's perspective. It was comprised of 4 multiple choice questions and 6 open-ended questions about their decision to continue to use the system, what they liked about the system, what they thought could be improved, and general comments.

4.4. COURSE ADMINISTRATOR SURVEY

Design an ER diagram in UML format for a game database given these details:

- A **publisher** is identified by **name** and **releases games**. A **game** may be **released** by multiple **publishers**, and each **publisher** gets credit for a **percent** of the game. (e.g. Publisher A 60%, Publisher B 40%).
- A **game** is identified by **id** and also has a **name**.
- A **game** has one **category**, and a **category** may describe multiple **games**. A **category** is identified by an **id** and has a **name**. A **category** may also have multiple subcategories.
- A **player** is identified by a **userid** and also has a **name** and **age**. A **player** may own multiple **games**, and a **game** may be owned by multiple **players**.
- A **session** is a particular **player** playing a **game**, and the **start time** is used to identify between **sessions** of the same **game** and **player** combination.
- During a **session**, **stats** are recorded. For a particular **session**, the **name** of the stat identifies it, and a **value** is stored for this **statistic**.

(a) Final Exam Question



(b) Final Exam Question

Figure 4.2: Final Exam Question and Answer

Chapter 5

Results

The results from the evaluations of the AutoER system are broken into two sections, System Evaluation and Student Performance Evaluation. In Section 5.1, the evaluations are focused on perceived system usability and student system preference. Section 5.2 focuses on student academic performance and submission behaviour patterns within the system. Common metrics used in the evaluations and their descriptions are listed in in Table 5.1.

Metric	Description
Users	The count of the users interacting with the AutoER system.
% of Participants	The percentage of total participants per study.
Submits	The average number of submissions per user to the AutoER system.
AutoER Grade	The average top grade of diagrams created and evaluated with AutoER.
Grade	The average grade of all participant's diagrams including grades of diagrams created with AutoER and all other methods.
Regressions	The average number of grade regressions per user. A grade regression occurs when a student submits a diagram that is graded lower than the previous attempt.
Seconds	The average number of seconds between submission to the AutoER system per user.
First Mark	The average grade of the first submission to the AutoER system.
% of Users	The percentage of total users interacting with the AutoER system.

Table 5.1: Description of Common Metrics in AutoER Evaluations

5.1 System Evaluation

5.1.1 System Use

The first analysis looked at if students prefer using AutoER compared to paper and other design software. The summary of usage for the two assignments and midterm and final exams is in Table 5.2.

	Assign 1		Assign 2		Midterm		Final	
	S	F	S	F	S	F	S	F
Users	30	141	27	130	43	172	41	172
% of Participants	67%	78%	61%	72%	98%	96%	93%	96%
Submits	83	32	108	37	25	5	57	6

Table 5.2: AutoER Usage Metrics

Students collaborated in groups for both Assignment 1 and 2. As a result, the number of system users for the assignments was lower than the number of users during the examinations. This is because groups of students would interact with the system using a single student account. Table 5.3 shows the number of students that uploaded diagrams created with AutoER as their final answer compared to the number of students that uploaded diagrams created by hand or with other systems. The number of students using AutoER was determined by counting all students in a group if the group submitted an answer created with AutoER.

Most students used AutoER for the first assignment, with an 89% upload rate in the summer and an 87% upload rate in the fall. AutoER does not support mapping to the relational model as required for the part of the second assignment, so usage was expected to be lower. Interestingly, the usage was still high as students appreciated checking their design model before adding data types and conversion. The average number of submissions for a

	Assign 1		Assign 2	
	S	F	S	F
Created with AutoER	39	157	34	170
% of Participants	89%	87%	77%	94%
Created with Other Methods	2	14	10	4
% of Participants	5%	8%	23%	2%

Table 5.3: Assignment Upload Metrics

question per user was high, and individual student traces indicated students used the feedback to improve their understanding and performance on the questions that would not have happened with a single evaluation point. This was also common for the large ER diagram in assignment 2.

98% of students in the summer and 96% in the fall used the AutoER system for the database design midterm exam question. A higher number was expected given that the system generated the question. Even though students did not have to use AutoER to answer the question, almost all did for reasons of efficiency and marking feedback.

Although the final exam question was an instructor-defined question and students were given the choice to complete this question manually or to use any diagram software, this question saw a similar usage to the midterm, with 93% and 96% usage in the summer and fall, respectively. This indicates the students' strong preference for the software and the ability to submit multiple times and have their answers automatically evaluated during the exam. Note that the number of submissions allowed during the exam depends on the policy enforced. This is discussed in detail in Section 5.2.5.

5.1.2 System Usability

The next analysis evaluated the system's usability and efficiency. The responses to the SUS questions are in Table 5.4. A SUS score reflects a user's subjective ranking of a system's usability. A score greater than 70 indicates a system is moderately user-friendly [BKM08]. AutoER had a SUS score of 77.01 in the summer study and a SUS score 76.95 in the fall study, showing that the system had good usability.

5.1.3 Free-form Survey Results

The free-form survey answers had a relatively small sample size and varied in depth of detail and because of this, the coding of feedback was deemed unnecessary. Instead, a subjective and qualitative analysis was performed to identify perceived strengths of the tool and to help guide improvements to the system.

The feedback contained many positive comments about the ability to build a diagram by interacting with the question text rather than drawing directly. Students indicated this helped them understand how to extract key concepts and add them to the model (which is a critical aspect of the design task). The majority of the feedback involved positive comments on AutoER's ease of use, which is a critical factor for learning design software

5.1. SYSTEM EVALUATION

Question	SD		D		N		A		SA	
	S	F	S	F	S	F	S	F	S	F
I think I would like to use this tool frequently.	4%	2%	0%	4%	8%	16%	27%	38%	62%	40%
I found the tool unnecessarily complex.	50%	32%	31%	56%	12%	8%	0%	2%	8%	2%
I thought the tool was easy to use.	0%	2%	4%	0%	4%	10%	35%	46%	58%	42%
I think that I would need the support of a technical person to be able to use this system.	65%	54%	12%	34%	8%	4%	0%	4%	15%	4%
I found the various functions in this tool were well integrated.	4%	2%	8%	6%	19%	10%	35%	50%	35%	30%
I thought there was too much inconsistency in this tool.	31%	24%	31%	52%	15%	12%	8%	8%	15%	2%
I would imagine that most people would learn to use this tool very quickly.	4%	2%	4%	2%	4%	10%	27%	44%	62%	42%
I found the tool very cumbersome to use.	35%	26%	31%	58%	19%	4%	4%	6%	12%	2%
I felt very confident using the tool.	0%	2%	12%	4%	12%	4%	31%	50%	46%	38%
I needed to learn a lot of things before I could get going with this tool.	54%	16%	15%	50%	15%	22%	8%	6%	8%	6%

Table 5.4: System Usability Scale survey results
Categories - SD: Strongly Disagree, D: Somewhat Disagree, N: Neutral, A: Somewhat Agree, SA: Strongly Agree

[ALS19]. Students also commented that the marking feedback helped them learn from and fix their mistakes, and several students indicated a desire for more detailed feedback when practicing.

A selection of quotes from students when asked what they found useful in the software include:

- “It was easy to use, especially in exams.”
- “It really helped me grasp the content.”
- “It was a good way to get feedback on diagrams.”
- “I think having AutoER is extremely useful and beneficial to students as it is helpful being able to interact with the diagram and receive real time feedback.”

Areas for improvement included the display of the diagrams, especially positioning of cardinalities on relationships, improved display of recursive relationships, the ability to add other types of relationships, and the ability to re-arrange diagrams.

A selection of quotes from students when asked what additional features they would like to see in the system:

- “I think it would be helpful to be able to drag the relationships into different order. This would make it easier to get a better looking diagram.”
- “Sometimes the cardinality gets harder to read as the number of entities increase. If this is solved that’ll be great!”
- “A better implementation of the way the relationships link up, a lot of the time the multiplicities are actually hidden behind entities. Other times when you make a relationship between 2 items the relationship line freaks out.”
- “More tools for testing relational algebra, sql, json, xpath.”

5.1.4 Administrator Evaluation

The survey for course administrators indicated that the software was found to be very useful and easy to use by participants. Administrators also commented positively about the time saved because of the grading precision of the software.

5.2. STUDENT PERFORMANCE EVALUATION

Reflecting similar feedback from the student survey, administrators generally suggested improvements in the display of the diagrams. They also would like the ability to search student attempts and a more robust marker feedback page.

5.2 Student Performance Evaluation

There were several metrics evaluated including the percentage of students using the software, the mean number of submissions per user, average time between submissions, and average final mark. A summary of the performance metrics is in Table 5.5.

Student performance was compared to the course offering in winter 2020 with 180 students that used the same instructional content, assignments, and final exam, but was taught by a different instructor.

	Assign 1		Assign 2		Midterm		Final	
	S	F	S	F	S	F	S	F
Users	30	141	27	130	44	172	41	172
AutoER Grade	91%	89%	76%	71%	95%	94%	91%	69%
Grade	93%	92%	94%	96%	93%	92%	88%	75%
Regressions	18.39	6.33	22.13	6.57	3.43	0.41	10.24	1.22
Submits	83	32	108	37	25	5	57	7
Seconds	1445	17276	808	10446	120	N/A ²	327	804

Table 5.5: AutoER Performance Metrics

5.2.1 Assignment Performance

The 2020 offering of COSC 304 had performance on both assignment 1 and 2 of 85%. The performance using AutoER was higher for both assignments in both studies. Only a portion of assignment 2 involves designing ER diagrams with the rest of the assignment requiring mapping to the relational model. Assignment 2 was the first lab for the project involving designing and building a database, so it is less predictive of AutoER’s usability.

This course was taught by a different instructor, creating a different instructional environment, which prevents making a conclusive statement regarding student performance. The number of assignment submissions created with other methods was not sufficient for comparison with submissions

²The average time between submissions per user has been removed from the analysis due to a deployment error.

created with AutoER. However, it is evident that student performance was equivalent or slightly improved from 2020, as students actively engaged with the system through multiple submissions, and survey feedback indicated a strong preference for using AutoER compared to Astah or other UML tools used previously.

5.2.2 Midterm Generated Question Performance

Performance on the generated midterm question was high, with an average grade of 93% and 92% on the summer and fall midterms, respectively. In the 2020 offering of the course, there was no comparable design question on the exam, however, the average grade from the 2019 COSC 304 midterm question was 70%. Unfortunately, the summer and fall midterms are also not directly comparable with the question from the 2019 midterm as that question was an instructor defined paper exam question.

The effectiveness of the auto-generated questions and marking were also informally evaluated by TAs after manually marking each midterm question submission. They noted that each student's answer requires about two minutes to grade manually when submitting on paper or using other software, whereas the AutoER tool did it instantly.

The AutoER system's auto-marking eliminated bias, mistakes, and variation between TAs which often occurs for large classes. The system provides detailed marking feedback to the TAs to verify correctness. This feedback can also be provided to students after the exam is complete.

5.2.3 Final Exam Performance

The final exam question used in both studies was an instructor-defined question that had been used in the previous offering of the course. In that previous offering, the average grade on the final exam question was 74%. The summer study's final question average grade was 88% which was an increase of 14% from the previous offering. The performance on the fall final question was significantly lower at 75%, though still slightly higher than the 2020 offering.

It is difficult to compare the final exam question results between the summer and fall studies as there were substantial differences in exam environments. In addition to the enforced submission policies not present in the summer, the students were only given 2.5 hours to complete the fall final exam versus the summer exam's 3-hour time limitation. The mark weightings were different between the exams as a result of this.

5.2.4 Summer 2021 Study

Further data analyzed relates to student performance and submission behaviour during the exam. In the summer study, students were allowed unlimited submissions to measure how they interact with the system with no restrictive submission policies enforced. The initial profile analysis compared the average number of submissions per question to the top grade achieved.

Figure 5.1 displays student performance by number of submissions with quartile groupings by total number of submissions during the midterm exam. First quartile students submitted the lowest number of times and achieved near perfect marks. These students solved a substantial amount of the problem before checking their answers. The fourth quartile of students had at least 31 submissions and demonstrated some undesirable behaviour such as trial-and-error. In the diagram, the dot represents the cut-off point of number of submissions to be in that quartile. The graph line continues to the end to make it easier to compare across quartiles. For example, the cutoff for the first quartile is 13 submissions.

Although categorizing students by their average number of submissions per question showed interesting results, it was determined that the average number of regressions (grade from submission decreases from last submission) per question better classified the students into submission behaviour profiles.

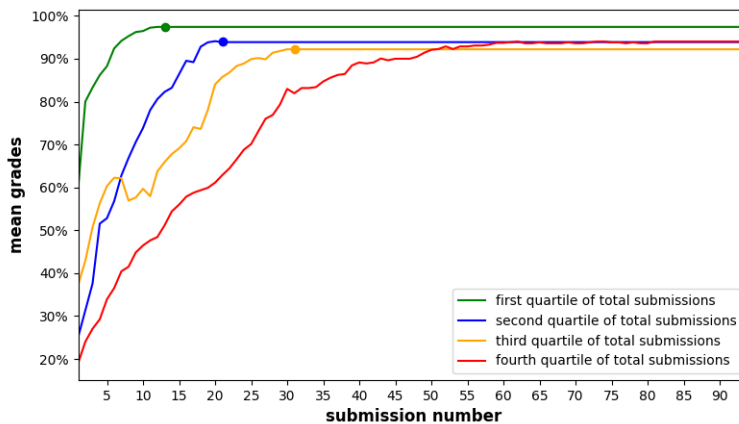


Figure 5.1: Student Performance Grouped By Total Submissions. Marker shows maximum quartile submissions cutoff.

5.2. STUDENT PERFORMANCE EVALUATION

Student Profiles

There are several distinctive student profiles (see Tables 5.6 and 5.7) that are visible in the data, and sample traces for each profile are shown in Figure 5.2. The profiles are distinguished by the number of regressions that were present in their answers. Note that although students are categorized by number of regressions, during the summer study, regressions were not displayed to the user and did not limit how the student answered the question. The regressions were calculated during post-analysis. While these categories are based on the number of regressions present in the submission history, other behavioural patterns emerged from the analysis. Most notably, some students demonstrated behaviours like nearly completing the question before the first submission, iterative development, and others showing less desirable trial and error exploration.

Category	Midterm %	Final %	Description
1	25.0	14.6%	No regressions
2	25.0	17.1%	1 to 2 regressions
3	25.0	17.1%	3 to 5 regressions
4	25.0	51.2%	6+ regressions

Table 5.6: Student Profile Categories

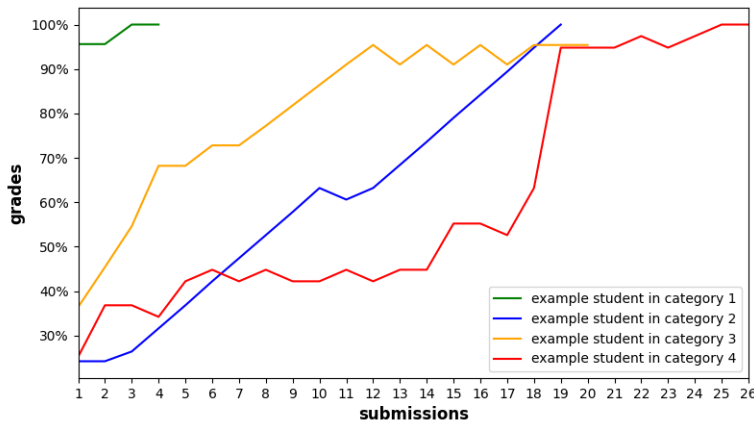


Figure 5.2: Example Student Performance Profiles

The top students in Category 1 achieved high scores with few attempts

5.2. STUDENT PERFORMANCE EVALUATION

	1		2		3		4	
	Mid	Final	Mid	Final	Mid	Final	Mid	Final
Users	11	6	11	7	11	7	11	21
AutoER Grade	99.6%	99.4%	95.0%	81.4%	92.5%	93.3%	91.5%	91.0%
First Mark	53%	48%	22%	24%	32%	11%	34%	14%
Regressions	0.00	0.00	1.45	1.29	4.00	4.14	8.27	18.19
Submits	8	12	17	17	30	37	45	90
Seconds	273	1854	68	257	73	103	79	62

Table 5.7: Summer Student Profile Metrics

with one student achieving 100% on the first attempt on the midterm, and two students achieving 100% on the first attempt on the final exam. These students would have similar performance if the number of attempts were reduced, or regression penalties were introduced [BZHH21]. Students in Category 2 have a low number of regressions, however, they may or may not have a low number of submissions. As shown in Figure 5.2, the example student in Category 2 used iterative development and feedback from the auto-grader to verify they are on the right track, and only had one regression. In Category 3, more regressions are present, which typically occur near the middle and the end of the problem. For the database design problem, identifying entities and attributes is the logical first step, and regressions in the middle may have allowed students to use trial-and-error along with the marking feedback to finalize the entities first. Regressions near the end seem to indicate that students may be using the same trial-and-error behaviour to find small mistakes (often missing relationships and incorrect cardinalities) before completing the question.

Category 4 students demonstrate numerous regressions throughout their attempts, in no particular area of the problem. This seems to indicate these students are using trial-and-error to solve the entire question rather than relying on course knowledge. All the highest submitting students were in Category 4.

Overall, the data and student profiles from the summer study validate research in [BZHH21] on reducing the number of submissions or introducing regression penalties to limit guessing. This motivated the development of the limiting attempts constraints for the system introduced in the fall study.

5.2. STUDENT PERFORMANCE EVALUATION

	Midterm			Final		
	S	F	Chg	S	F	Chg
Grade	93%	92%	-1%	88%	75%	-13%
Regressions	3.43	0.41	-88%	10.24	1.22	-88%
Submits	25	5	-82%	57	7	-88%
Seconds				327	804	146%

Table 5.8: Limiting Attempts Change

5.2.5 Fall 2021 Study

Submission Policy Impact

The first study revealed that some of the students that used the AutoER system in the summer seemed to be utilizing trial-and-error for solving questions. In response to that, submission policies as discussed in [BZHH21] were introduced to the system for the fall study. These policies were only enforced for the AutoER examination questions. For the fall midterm question, the students were assigned either the regression penalty or were limited to 7 attempts on the question. For the fall final exam, they were given the option of choosing either of the two policies.

Table 5.8 shows an overview of the changes to the submission behaviour of the users of the system. The number of average grade regressions decreased by 82% and the number of regressions dropped 88% on the fall midterm. The regressions and submissions were reduced by 88% and the average seconds between submissions per user was increased by 146% on the final exam question.

These changes demonstrate that students in the second study were more thoughtful about how they approached submitting answers, ensuring that the answer is as accurate as possible before submission rather than repeatedly checking and changing their answers during development. This indicates that adding limitation constraints positively affect how users approached solving problems in the system.

Submission Policy Impact By Profile

The data was categorized into the same regression profiles as examined in the summer study for comparison. The result of the analysis is shown in Table 5.9. One of the most obvious changes between the studies is that the percentage of students in Category 1 (0 regressions) was increased by over 180% for both the midterm and final fall exam. The number of students in

5.2. STUDENT PERFORMANCE EVALUATION

Category 1						
	Midterm			Final		
	S	F	Chg	S	F	Chg
Users	11	127		6	69	
% of Users	25.0%	73.8%	195.3%	14.6%	41.3%	182.1%
AutoER Grade	99.6%	95.2%	-4.4%	99.4%	69.4%	-30.1%
First Mark	52.6%	81.2%	54.3%	48.3%	53.6%	11.0%
Regressions	0.0	0.0	0.0%	0.0	0.0	0.0%
Submits	8	3	-65.1%	12	3	-73.1%
Seconds				1854	1211	-34.7%
Category 2						
	Midterm			Final		
	S	F	Chg	S	F	Chg
Users	11	40		7	80	
% of Users	25.0%	23.3%	-7.0%	17.1%	46.5%	172.4%
AutoER Grade	95.0%	92.4%	-2.7%	81.4%	71.7%	-11.9%
First Mark	21.9%	70.7%	222.3%	23.5%	49.2%	109.3%
Regressions	1.45	1.20	-17.5%	1.29	1.41	9.9%
Submits	17	7	-57.3%	17	6	-63.4%
Seconds				257	589	129.3%
Category 3						
	Midterm			Final		
	S	F	Chg	S	F	Chg
Users	11	4		7	20	
% of Users	25.0%	2.3%	-90.7%	17.1%	11.6%	-31.9%
AutoER Grade	92.5%	82.7%	-10.6%	93.3%	61.3%	-34.3%
First Mark	32.4%	37.9%	17.0%	11.1%	36.3%	225.5%
Regressions	4.0	4.0	0.0%	4.1	3.5	-15.5%
Submits	30	26	-13.8%	37	14	-62.0%
Seconds				103	360	247.5%
Category 4						
	Midterm			Final		
	S	F	Chg	S	F	Chg
Users	11	1		21	3	
% of Users	25.0%	0.6%	-97.7%	51.2%	1.7%	-96.6%
AutoER Grade	91.5%	87.2%	-4.7%	91.0%	62.6%	-31.3%
First Mark	34.1%	67.8%	98.9%	14.0%	28.2%	101.8%
Regressions	8.3	7.0	-15.4%	18.2	8.7	-52.4%
Submits	45	25	-44.3%	90	38	-57.3%
Seconds				62	104	68.9%

Table 5.9: Comparison of Profiles Across Studies

5.2. STUDENT PERFORMANCE EVALUATION

Category 2 (1 to 2 regressions) was increased by over 170% for the final fall exams. The users that were classified into these categories during the first study showed desirable behaviours in their approach to solving questions in the system. Similarly, most users in these top categories in the fall seem to use strategies that demonstrate topic knowledge rather than relying on guessing.

When examining the individual performances in Category 2, there seems to be a behaviour change in that most users seem to no longer be using iterative development but are behaving more similarly to a Category 1 users from the summer, demonstrating a high first mark with few submits. The only difference between the two categories in the second study is that there are one or two grade regressions present in a Category 2 user's answer history. Sample traces for each profile in the fall study are shown in Figure 5.3.

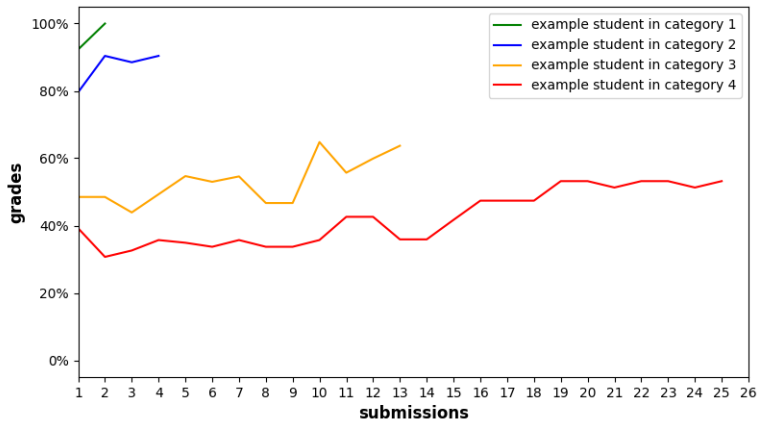


Figure 5.3: Example Fall Student Performance Profiles

The positive changes were not limited to the top categories as users in every category show improvement in some metrics. All categories show a decrease in the average number of submissions and an increase to the average of the first mark submitted, as shown in Table 5.9. The decrease in Category 4 (6+ regressions) users is the most substantial, with a 98% decrease on the midterm and a 97% decrease on the final exam. However, upon examination of individual submissions, the trial-and-error behaviour seems to be still visible in the submission history for a few students.

The individual submission history for users in Category 3 (3 to 5 re-

5.2. STUDENT PERFORMANCE EVALUATION

gressions) tend to display an iterative development procedure, like the first study. However, they also seem to display a more randomized regression pattern indicating the possibility of more guessing at answers rather than relying entirely on prior knowledge.

One surprising finding was that Category 1 users' final exam grade averages were lower than Category 2 users in the second study. Upon more in-depth analysis, there was a new behaviour not noted in the first study's examinations: an increase of one attempt submission histories. The analysis of one-attempt submissions is shown in Table 5.10. This behaviour could be a result of the 30-minute decrease in time given for the final exam with a similar number of questions. This increase of one-attempt submission histories lowers the overall grade average of users in Category 1 from 70.0% to 69.4%, however, it does not entirely explain the finding as the grade average from the Category 2 users is 71.7%.

	Users	AutoER Grade
Summer Midterm	1	100%
Summer Final	1	100%
Fall Midterm	45	96.3%
Fall Final	19	68.1%

Table 5.10: User Count and Grades with One Submission

Overall, there is strong evidence that limiting attempts in the system alters user behaviours in positive ways, from reducing grade regressions and number of submission attempts to increasing time spent on answers.

Comparing Submission Policies

The submission policies attempt to solve the problem of over-reliance on the automatic marker with different approaches. The maximum attempt limitation allows the instructor to set a maximum number of attempts on a question, while the regression penalty imposes a permanent cost on submitting answers that are graded lower than the previous submission.

The data was analyzed to examine which of the two submission policies is more effective at limiting trial-and-error behaviour with the least impact on student performance. Comparisons of means from the collected data were performed using two-sample t-Tests assuming unequal variances with significance level $\alpha = 0.05$, and the p-values are provided.

Table 5.11 shows that the difference between the midterm grades was not significant ($p = 0.84$) between the two strategies, but the final exam

5.2. STUDENT PERFORMANCE EVALUATION

data shows that students who had the maximum attempts submission policy achieved a grade that was nearly 10% higher ($p = 0.04$). The students who used the maximum attempt policy also have a statistically significant lower average submits and longer submit times, possibly indicating that those users spent more time taking into consideration the quality and quantity of their answers before submission.

	Midterm		Final	
	MaxAtt	Regr	MaxAtt	Regr
Users	50	122	137	35
AutoER Grade	95%	94%	71%	62%
Regressions	0.320	0.451	0.934	2.314
Submits	3	5	5	14
Seconds			854	607

Table 5.11: Performance Comparison of Submission Policies

Interestingly, the number of regressions is higher when the question is marked with a regression penalty. This makes sense for several reasons. If there is a limited number of attempts, then there are a limited number of regressions that can happen. It also stands to reason that if the user knows that they have an unlimited number of submissions, they may be more likely to submit with a trial-and-error strategy. The fact that the average number of submissions and the average number of seconds between submissions was higher for the users with the maximum attempts policy also support this argument.

Submission Policies Across Student Profiles

The average grade of students in each category was lower for the regression penalty compared to maximum attempts limitation. The first submission mark was generally considerably lower as well. Pie charts comparing the number of students per category for unlimited submissions, maximum attempts policy, and regression penalty for the midterm and final exams are in Figures 5.4 and 5.5 respectively.

Examining the submission policies over student profiles further illustrates the maximum attempts restriction's effectiveness. Table 5.12 shows the comparison of submission policies across student profiles. The average first submission mark was higher in Category 1 and Category 3 with the maximum attempt limitation. It varied between exams in Category 2 with

5.2. STUDENT PERFORMANCE EVALUATION

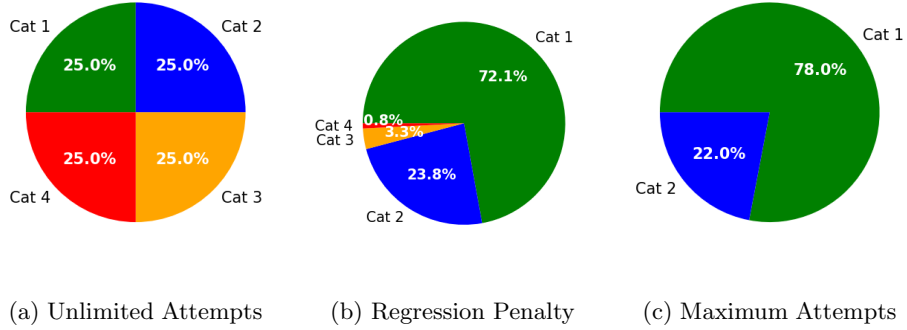


Figure 5.4: Midterm Exam Students by Category and Submission Policy

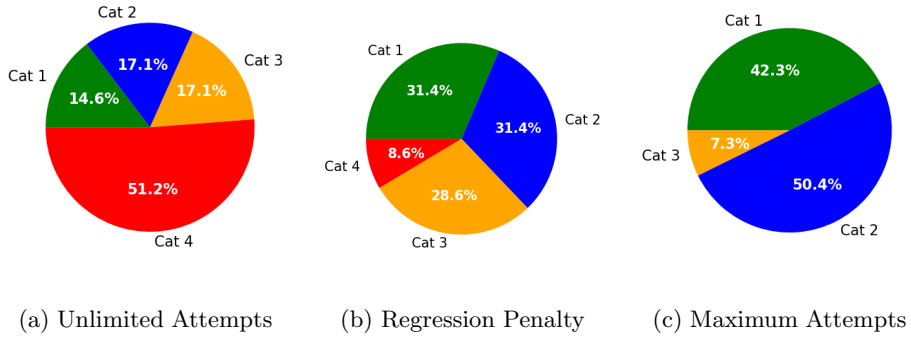


Figure 5.5: Final Exam Students by Category and Submission Policy

10% lower average grade for the first attempt on the midterm and 20% higher first grade on the final exam. The average seconds between attempts in Category 2 were also lower, while the average time was higher for both Category 1 and Category 3. Overall, most measures favour the maximum attempt restriction over the regression penalty.

The maximum attempt policy with the restriction of 7 submissions also prevented any user from being classified as Category 4 with either the midterm or final exam. This is because it would be improbable for a user to have 6 regressions with a maximum of 7 submissions.

Visualizations of student submission profiles are in Figures 5.6 and 5.7. The lines are colored based on the category the student was assigned to (category 1 - green, category 2 - blue, category 3 - yellow, and category

5.2. STUDENT PERFORMANCE EVALUATION

	Category 1		Category 2		Category 3		Category 4	
	MaxAtt	Regr	MaxAtt	Regr	MaxAtt	Regr	MaxAtt	Regr
Midterm								
Users	39	88	11	29	0	4	0	1
AutoER Grade	95%	95%	93%	92%		83%		87%
First Mark	92%	77%	64%	73%		38%		68%
Regressions	0.00	0.00	1.45	1.10		4.00		7.00
Submits	2	3	6	8		26		25
Final								
Users	58	11	69	11	10	10	0	3
AutoER Grade	70%	66%	73%	61%	63%	59%		63%
First Mark	55%	44%	52%	32%	52%	20%		28%
Regressions	0.00	0.00	1.41	1.45	3.10	3.90		8.67
Submits	3	4	6	10	7	21		38
Seconds	1249	1014	584	621	424	296		104

Table 5.12: Limiting Attempts Across Student Profiles

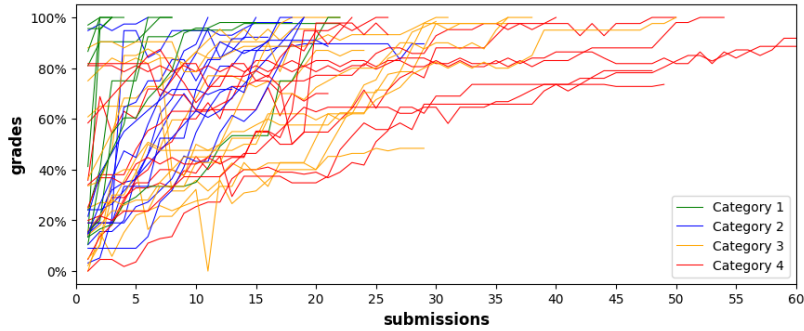
4 - red). Note that the Figure 5.6a has been reduced from showing 91 submissions to showing a maximum of 60 and Figure 5.7a has been reduced from 199 to 120. An obvious impact of the submission policy is the dramatic reduction in the number of submissions. There was a hard cutoff of seven submissions for the maximum attempt policy, and many students did not reach that cutoff. Although the regression penalty reduces the number of submissions, there was still a percentage of students with a high number of submissions (and regressions). There were more regressions for students that had a regression penalty versus a limited number of submissions. This is an interesting result as the regression penalty on the student’s grade was only applied for students in the regression penalty submission policy. The results were similar for both exams. The final exam with submission penalties had a much more significant reduction in submissions and regressions.

Student Preference

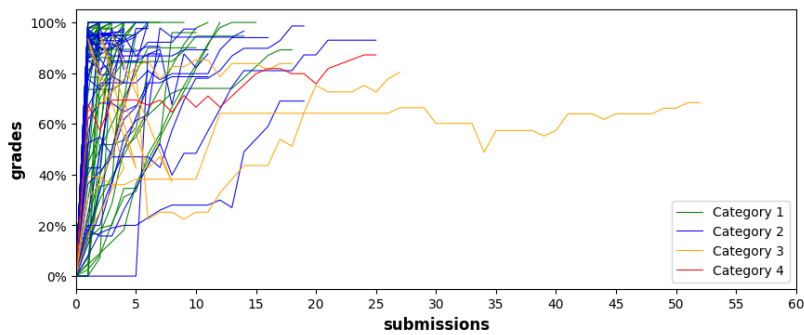
The analysis examined what the fall students chose for their final exam submission policy given the policy that they were assigned for the midterm. The results are shown in Table 5.13. Overall, given the choice of the maximum attempts restriction or the regression penalty, 80% of students chose the maximum attempt policy. The students that chose this for the final exam did 10% better overall on the exam question in the AutoER system. Interestingly, students that chose this option for the final exam also tended to do better in the course, scoring 2% higher on the randomly assigned, automatically generated midterm question, and 3% higher overall.

Further analysis revealed that 74% of users that were assigned the

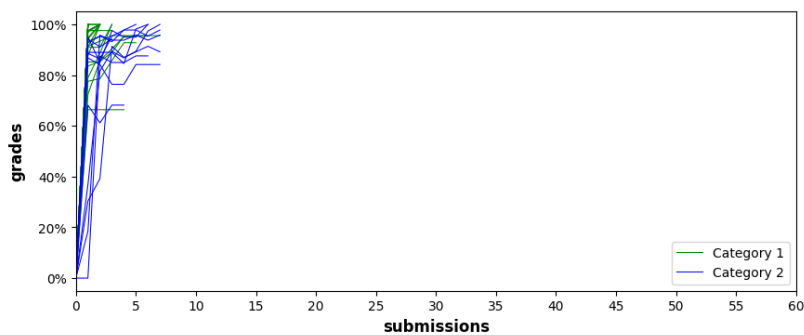
5.2. STUDENT PERFORMANCE EVALUATION



(a) Unlimited Submissions



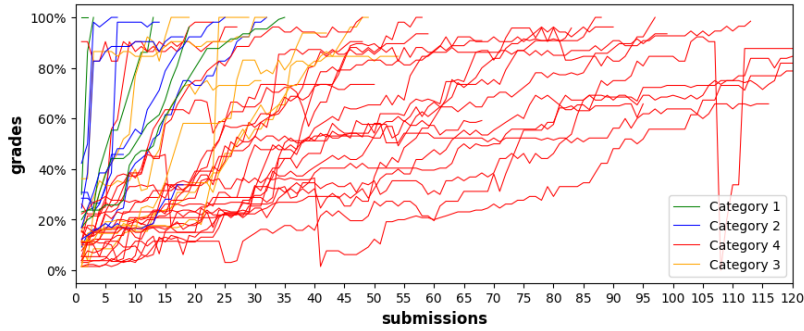
(b) Regression Penalty



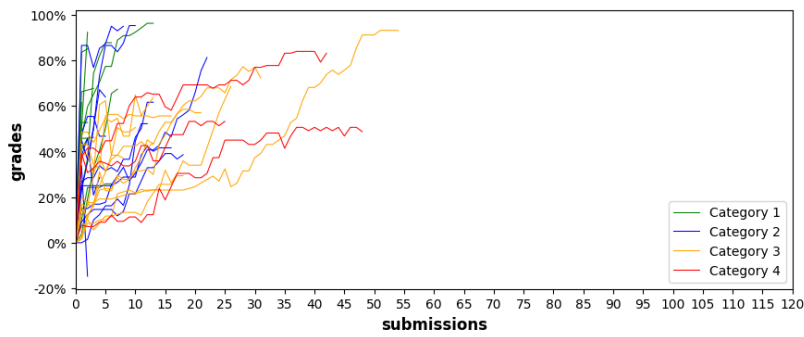
(c) Maximum Attempts

Figure 5.6: Midterm Exam Students by Category and Submission Policy

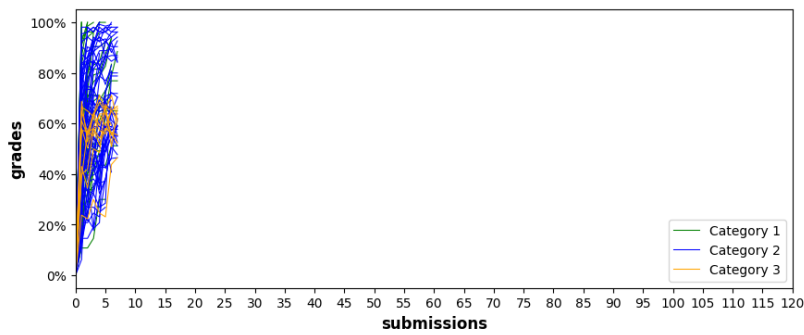
5.2. STUDENT PERFORMANCE EVALUATION



(a) Unlimited Submissions



(b) Regression Penalty



(c) Maximum Attempts

Figure 5.7: Final Exam Students by Category and Submission Policy

5.3. DISCUSSION

regression penalty during the midterm chose to use maximum attempts limitation on the final exam. Nearly all the users that were assigned the maximum attempts policy on their midterm chose that same policy on the final exam. It appears that although users could have unlimited attempts with feedback on a final exam question with a regression penalty, the maximum attempts submission policy may have been considered to be the safer option as it doesn't penalize the overall marks for mistakes along the way.

	Users	% of Users	Mid	Final	Dif	Course
Regr Choice	34	20%	92%	61%	-31%	81%
Regr ->Regr	31	18%	92%	61%	-32%	81%
MaxAtt ->Regr	3	2%	89%	65%	-24%	79%
MaxAtt Choice	137	80%	94%	71%	-23%	84%
Regr ->MaxAtt	90	53%	94%	70%	-24%	83%
MaxAtt ->MaxAtt	47	27%	95%	73%	-21%	85%
Total	171	100%	94%	69%	-25%	83%

Table 5.13: Limiting Attempts: Student Preference and Performance

5.3 Discussion

The AutoER system is a simple pedagogical tool designed to reinforce learned concepts in an introductory database design course. For a learning tool to be deemed effective, there must be some motivation for students to choose to use the system over traditional methods. The first research question (**RQ1**) examined whether students preferred to use the AutoER system compared to paper or other existing design software. Because of the unique features of AutoER, there was no direct experimental comparison with other existing software, however, students were given the option to use paper or any other system for all database design questions. The usage data demonstrates that the students preferred using the AutoER system compared to other options with a high number of users for all assignments and examination questions.

Another metric to gauge a learning tool's effectiveness is the perceived usability of the system. A usable system is easy to learn and allows the student to focus on the task at hand, rather than having to spend time familiarizing themselves with the tool itself. AutoER's usability was explored with the second research question (**RQ2**). The system usability survey [Bro96] completed by the students revealed that the system had SUS scores of around 77% in both studies. The free-form comments were also

5.3. DISCUSSION

positive, highlighting that interacting with text rather than drawing helped them learn concepts, especially with feedback. Although there were areas for improvement noted, such as issues with the display of relationships of the diagram, overall, the survey results indicate that the system has good usability and that the students perceived the tool to be effective and helpful.

The third research question (**RQ3**) investigated if the AutoER system's automatic grading approach increased student performance on assignments. In both studies, student performance on assignment questions was higher than the previous offering. The average assignment grades were in the range 92-96% compared to about 85% in the 2020 fall offering of the course. However, due to the differences in instructional environments, a definitive answer could not be reached. Although there was evidence of increased performance and high system usage, further study is required to make a conclusive statement.

The question auto generation capability is a unique feature not available in any other system to the author's knowledge. Although question generation in general has been extensively researched, the domain-specific area of database design diagrams has been overlooked. **RQ4** investigated whether this novel feature was effective for use in exam evaluations. More than 95% of students chose to use AutoER for the midterm exam question in both studies. Teaching assistants manually validated each question generated and the associated auto-marking, and they noted that there were no issues with unsolvable questions or marking that was inconsistent with established guidelines. They also remarked that each student answer requires about two minutes to grade manually when submitting on paper or using other software. The auto-marking saved time as the evaluation is immediate, and it eliminates bias, mistakes, and variation between TAs which often occurs for large classes. This indicates that AutoER's question generation feature and auto evaluation approach is effective for use in examinations.

The data from the first study revealed some distinctive behaviour patterns in the submission history of the users. These users were categorized into 4 profiles based on the number of regressions in the answers. Students in the Category 1 displayed positive submission behaviours such as high marks on their first submission and a low number of overall submissions. Users in Category 4 had a higher number of average submissions and regressions throughout their history, potentially demonstrating trial-and-error behaviour. To attempt to limit this type of reliance on the auto-grader, attempt limiting submission policies were added to the system for the second study. The hypothesis that student submission behavior would be positively affected by submission limiting policies is confirmed by the collected data.

5.3. DISCUSSION

Both strategies of either using a maximum number of submissions or a regression penalty are effective at reducing the number of submissions and regressions by 82% to 88% and increasing the time between submissions by 146%. There is strong evidence supporting **RQ5** on improving student behaviors when interacting with automatic assessment systems. Specifically, the amount of trial-and-error behavior is greatly reduced.

Comparing students based on regressions rather than submissions is a better approach, regardless if the system penalizes students for regression mistakes. Since a regression occurs when a student's mark goes down, poor student behaviors of trial-and-error, exploiting system feedback, and random guessing are much more likely when a student makes a mistake. If a student has multiple submissions, but the grade continues to increase, it is an interesting debate whether that is a poor behavior that should be prevented. However, many regressions clearly indicate a pattern of not understanding the question sufficiently to improve the mark after every submission. The number of regressions is closely related with the number of submissions and is a better indicator of student mistakes.

The sixth research question (**RQ6**) was designed to identify what submission limiting policy is the most effective. Although both maximum attempts and regression penalty show positive improvements overall, the maximum attempt policy data displayed higher student performance and better submission behavioural patterns. Student performance on the final exam was 10% higher with the maximum attempts limitation, and these students had lower number of attempts and regressions. There were students with the regression penalty limitation that still had a high number of regressions and submissions, which are behaviours that should be avoided.

High performing students excel on either submission policy as they use very few submissions with limited mistakes. As more submissions are performed, more regressions occur for all students. However, the bounded number of submissions results in students spending more time between submissions which decreases the chance of regressions.

However, it should be noted that the difference in behaviour and performance was more notable in the final exam, where students chose their policy, than in the midterm exam, where students were assigned their policy. This indicates the possibility of a selection effect, where more of the higher performing students may have chosen the maximum attempt policy and more of the weaker performing students may have chosen the regression penalty. It is possible that stronger-performing students may have viewed the possibility of a permanent penalty to their final mark as a riskier choice, while weaker performing students may have viewed unlimited attempts as

5.3. DISCUSSION

a better strategic choice. If this is the case, it could have lead to the higher performance and improved behavioural patterns viewed in the data for the maximum attempt policy compared to the regression penalty policy for the final exam.

When considering the load on the automatic assessment system, a maximum attempts policy is the clear winner as students make fewer submissions overall and the absolute number of submissions is bounded. These results are in-line with the results in [BZHH21] that studied the impact of regressions for reducing the load on the assessment system. The additional results comparing with maximum attempts and student preference are important contributions.

The final research question (**RQ7**) tested student preference for the submission policy. Given the choice on a high stakes final exam where their marks were directly impacted, the vast majority (80%) selected maximum attempts as their preferred submission policy. It appears the maximum attempts restriction may have been perceived to be less risky by the students as it doesn't permanently penalize the overall marks, however, there was no survey data collected to confirm this assumption. Although it can be definitively stated that the students prefer the submission policy of maximum attempts over the regression penalty, more research is required on the exploration of the motivation of the preference.

Overall, the results demonstrate that the AutoER system is an easy to use, effective database design learning tool with novel features. Allowing students to interact with question text directly to build diagrams ensures that the system is able to provide consistent, accurate, and immediate evaluation of diagrams regardless of class size. The ability to generate random domain-specific questions is also beneficial as it allows for the creation of unique variants of questions in an examination where the potential of academic dishonesty is high, particularly during online exams. The addition of attempt-limiting submission policies to the system encourages students to be thoughtful about their diagram creation and reduces over-reliance on the automatic grader. Of the two submission policies implemented in the system, limiting the number of attempts has shown to be more effective and preferred by students, although both policies have a positive impact on student behaviour.

Chapter 6

Conclusion

This thesis sought to demonstrate that utilizing the AutoER system in an introductory database course for UML design questions can support student learning and improve course administrators' efficiency by providing automated real-time feedback to students. A general background on the challenge of learning diagram creation and software for database design used in classrooms in database courses was explored in Section 2.1. Database design can be difficult for students at first as they must learn the proper syntax and learn to translate the written text into the appropriate diagram elements. Diagram creation can use ER or UML notation and can be drawn on paper or created with the help of diagram software. Surveys of UML software indicate the preference for easy-to-use systems focusing on learning rather than commercial products used by experts. Whether designed on paper or created with the help of software, consistent manual evaluation of the diagrams can be time-consuming, particularly for large classes, because students can use inconsistent naming and labels, can have trouble understand fundamental modeling constraints, and can misinterpret the question.

As discussed in Section 2.2, automatic diagram evaluation with real-time feedback allows for increased student engagement and increased consistency and efficiency in grading for course administrators. Automatic generation of questions allows practicing until mastery and for online exams to reduce academic dishonesty. Prior work examined automatic diagram marking using image processing, structural and semantic analysis, and machine learning. Although evaluation with some systems can be accurate to supplied answers within about 10%, challenges remain with achieving higher accuracy. Currently, there is no support for auto-generation of questions within these systems.

The AutoER system architecture and available features are described in Chapter 3. The web-based system allows students to build diagrams by interacting with question text directly rather than drawing diagram components manually. This improves speed and guarantees consistent naming for precise marking. As the student is building their answer, the system

continuously generates a visual representation of their answer.

AutoER supports both instructor-created and automatically generated questions and is flexible allowing modification to its user interface, marking and feedback, and question generation parameters. The marking process is precise as students use standardized names when building a diagram and the feedback from the auto-grader is provided to a student by request at any time. Randomized questions are generated given parameters on the number of entities, attributes, and relationships. The randomized questions have a single URL for use in a testing system, but each student gets their own question variant. To combat relying on the auto-grader to answer questions, two types of penalties that restricted attempts were added to the system: maximum attempts, which limits the number of submissions and a regression penalty, which allows unlimited attempts but applies a permanent penalty whenever a student's mark goes down.

The system was evaluated in two offerings of an undergraduate database course in 2021 at the University of British Columbia Okanagan. Presented in Chapter 4, the studies collected course data from two assignments, an automatically generated question on a midterm exam, and an instructor-defined question on the final exam. Students could use the AutoER for any of these questions or answer questions on paper or with other UML software previously used. The first study allowed for unlimited attempts on all questions. The second study randomly assigned either maximum attempt limitation or regression penalty submission policy on the midterm question and allowed the student to choose their submission policy on the final exam. After using the system for several weeks, students completed a usability survey and provided feedback (see Appendix B.1) regarding their choice to use the system.

The results of the studies were examined in Chapter 5. The usage and survey data demonstrates high student satisfaction compared to traditional UML design question formats and a preference for using the software to improve their learning outcomes. The generation of question and automatic evaluation proved to be effective for use in examinations with TAs noting that the system reduced time spent marking and ensured consistency and accuracy in evaluation.

Data from the first study revealed some distinctive behavioural patterns in the submission history of the users with some students demonstrating an over-reliance on the auto-grader denoted with a high number of submissions and grade regressions. To attempt to limit this type of behaviour, two types of submission policies were added to the system, a regression penalty and a maximum attempt limitation, for the second study. With the

submission policies enforced, the average number of regressions and submissions decreased significantly indicating that the users were being more thoughtful about their answers, relying on course knowledge rather than AutoER's evaluation feedback to answer questions. A comparison between the two submission policies revealed that student performance was higher and trial-and-error behaviour was lower when the students used the maximum attempt limitation. The data also revealed that the majority of students chose the maximum attempt submission policy when given an option.

6.1 Limitations and Threats to Validity

The research evaluation was designed to determine the usability of AutoER for question generation and evaluation. The SUS evaluation demonstrated good usability metrics. There was no comparator tool taught in the course or an evaluation with any other auto-grading UML software. Survey results demonstrate that the students perceived AutoER to be effective, but there is no data to determine how it compares to other auto-grading systems. Further, it is not possible to directly compare with such systems as AutoER allows students to build answers using consistent terminology rather than drawing the diagram and entering their own names. This approach to question answering ensures precise marking according to criteria unlike other approaches that have more matching ambiguity.

Although the system statistics indicate a high number of student submissions and strong overall performance on the assignments and exams, this data is insufficient to argue conclusively there is increased engagement and learning. Students clearly preferred using AutoER, and their grades were higher than the previous offering without it. However, the differences in instructional environments may have had an impact on this result.

To evaluate the impact of the AutoER system on learning and engagement, students from the same environment could be randomly separated into two groups for two assignments: For the first assignment, one group could be designated an experimental group utilizing the AutoER system for diagram creation and evaluation and the other a control group using traditional methods as a baseline. For the second assignment, the groups could swap roles. For each of the assignments, a pre-test and post-test could be conducted to measure the learning gains of the assignment. This could allow for a more conclusive assessment of learning with the AutoER system. However, this type of experimentation may offer an unfair advantage to some students and was deemed unsuitable for these studies.

6.2. FUTURE WORK

The evaluation of submission policies was designed to determine the effect of submission limits in realistic student experiences. The comparison between maximum attempts and regression was done within a single course section allowing for an identical instructional environment. Although the random assignment between the two submission limits on the midterm exam was not equal due to an error, there were sufficient numbers of students in both categories. Allowing students to select their approach on the final exam was fair and did not affect grouping in any way, however, it may have led to a selection effect, where higher performing students and lower performing students chose different policies based on their needs. The student self-selection data allows for statements on student preference in an actual situation, but without a survey after the final exam there is no data to determine why students selected one technique over the other.

Comparing the data collected from the second course section with submission limits with the first section with no limits is reasonable, with the acknowledgement that the two course sections are not identical instructional environments. Although the instructor and all materials were the same, the first course section had a smaller class size. Ideally, some students may have been randomly selected to have no submission limits in the second course section, but this was determined to be an unreasonable advantage compared to other students.

6.2 Future Work

Future work will extend the evaluation for more course offerings and integrate the system into learning management systems using the Learning Tools Interoperability (LTI) standard. The system will be improved by the addition of an instructor's frontend to allow for easier question creation. The question interface will also be improved to allow for more editing of the diagrams and improved visual representation. Although the system was tested for database design diagrams using UML, it is also applicable to other database diagram notations and UML diagrams for other use cases such as developing an object-oriented class diagram. Other planned work includes development of interfaces for the aforementioned use-cases, explorations of different types of feedback, and the examination of variants of the submission limits such as combining regressions and maximum attempts, and the exploration of different maximum attempt limits.

Further evaluation of student submission behaviour in discipline-agnostic auto-grading systems is also future work planned beyond the further devel-

6.2. *FUTURE WORK*

opment of the AutoER system. Proposed work includes researching the impact of different types of submission limitations on varying problem types across academic disciplines and exploring the correlation of performance and student submission policy preference.

Bibliography

- [AL17] Luciane Telinski Wiedermann Agner and Timothy C. Lethbridge. A survey of tool use in modeling education. In *20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, Austin, TX, USA, September 17-22, 2017*, pages 303–311. IEEE Computer Society, 2017. → pages 4
- [ALS19] Luciane Telinski Wiedermann Agner, Timothy C. Lethbridge, and Inali Wisniewski Soares. Student experience with software modeling tools. *Softw. Syst. Model.*, 18(5):3025–3047, 2019. → pages 4, 46
- [Auv15] Tapio Auvinen. Harmful study habits in online learning environments with automatic assessment. In *2015 International Conference on Learning and Teaching in Computing and Engineering*, pages 50–57, 2015. → pages 11
- [AV03] Carl Alphonse and Phil Ventura. Quickuml: a tool to support iterative design and code development. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2003, October 26-30, 2003, Anaheim, CA, USA*, pages 80–81. ACM, 2003. → pages 6
- [BAK19] Weiyi Bian, Omar Alam, and Jörg Kienzle. Automated grading of class diagrams. In *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019*, pages 700–709. IEEE, 2019. → pages 1, 9
- [BAK20] Weiyi Bian, Omar Alam, and Jörg Kienzle. Is automated grading of models effective? assessing automated grading of class diagrams. In *Proceedings of the 23rd ACM/IEEE International*

- Conference on Model Driven Engineering Languages and Systems*, MODELS '20, page 365–376, New York, NY, USA, 2020. ACM. → pages 9, 31
- [BE15] Kevin Buffardi and Stephen H. Edwards. Reconsidering automated feedback: A test-driven approach. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE 2015, Kansas City, MO, USA, March 4-7, 2015*, pages 416–420. ACM, 2015. → pages 1
- [BKM08] Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008. → pages 39, 44
- [BMM20] Younes Boubekeur, Gunter Mussbacher, and Shane McIntosh. Automatic assessment of students’ software models using a simple heuristic and machine learning. In *MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Canada, 18-23 October, 2020, Companion Proceedings*, pages 20:1–20:10. ACM, 2020. → pages 1, 8
- [Bro96] John Brooke. Sus: A quick and dirty usability scale. In *Usability Evaluation in Industry*, pages 189–194, 1996. → pages 39, 61
- [BZHH21] Elisa L. A. Baniassad, Lucas Zamprogno, Braxton Hall, and Reid Holmes. STOP THE (AUTOGRADER) INSANITY: regression penalties to deter autograder overreliance. In Mark Sherriff, Laurence D. Merkle, Pamela A. Cutter, Alvaro E. Monge, and Judithe Sheard, editors, *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, pages 1062–1068. ACM, 2021. → pages 10, 11, 22, 51, 52, 64
- [Che76] Peter P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976. → pages 4
- [CWZ18] Binglin Chen, Matthew West, and Craig B. Zilles. How much randomization is needed to deter collaborative cheating on

- asynchronous exams? In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale, London, UK, June 26-28, 2018*, pages 62:1–62:10. ACM, 2018. → pages 10
- [Edw04] Stephen H. Edwards. Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '04*, page 26–30, New York, NY, USA, 2004. Association for Computing Machinery. → pages 11
- [EP08] Stephen H. Edwards and Manuel A. Pérez-Quiñones. Webcat: automatically grading programming assignments. In *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008*, page 328. ACM, 2008. → pages 1, 7, 10, 14
- [FGF21] Emily Faulconer, J. C. Griffith, and H. Frank. If at first you do not succeed: student behavior when provided feedforward with multiple trials for online summative assessments. *Teaching in Higher Education*, 26(4):586–601, 2021. → pages 12
- [GM20] Carlos R. Jaimez González and Jazmín Martínez-Samora. Diagrammer: A web application to support the teaching-learning process of database courses through the creation of E-R diagrams. *iJET*, 15(19):4–21, 2020. → pages 6
- [Gro17] Object Management Group. Omg® unified modeling language® (omg uml®), 2017. Retrieved: 2021-08-13. → pages 4
- [Has11] Robert W Hasker. Umlgrader: an automated class diagram grader. *Journal of Computing Sciences in Colleges*, 27(1):47–54, 2011. → pages 8, 31
- [HBTN21] Georgiana Haldeman, Monica Babes-Vroman, Andrew Tjang, and Thu D. Nguyen. CSF: formative feedback in autograding. *ACM Trans. Comput. Educ.*, 21(3):21:1–21:30, 2021. → pages 11
- [HR11] Robert W Hasker and Mike Rowe. Umlint: Identifying defects in uml diagrams. In *2011 ASEE Annual Conference & Exposition*, pages 22.1558.1 – 22.1558.14, 2011. → pages 8

- [IAKS10] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, page 86–93, New York, NY, USA, 2010. Association for Computing Machinery. → pages 12
- [KKM06] Ville Karavirta, Ari Korhonen, and Lauri Malmi. On the use of resubmissions in automatic assessment systems. *Comput. Sci. Educ.*, 16(3):229–240, 2006. → pages 11
- [L⁺66] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966. → pages 9
- [Lee21] Haden Hooyeon Lee. Effectiveness of real-time feedback and instructive hints in graduate CS courses via automated grading system. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, pages 101–107. ACM, 2021. → pages 7
- [LSTA21] Madeleine Lorås, Guttorm Sindre, Hallvard Trøttestad, and Trond Aalberg. Study behavior in computing education—a systematic literature review. *ACM Trans. Comput. Educ.*, 22(1), oct 2021. → pages 10
- [MBF⁺18] Marina Marchisio, Alice Barana, Michele Fioravera, Sergio Rabbellino, and Alberto Conte. A model of formative automatic assessment and interactive feedback for stem. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 1016–1025, 2018. → pages 11
- [MDP20] Samiha Marwan, Anay Dombe, and Thomas W. Price. Unproductive help-seeking in programming: What it is and how to address it. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 54–60, New York, NY, USA, 2020. Association for Computing Machinery. → pages 11
- [MGF⁺20] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. Adaptive immediate feedback can improve novice programming engagement and intention to persist in

- computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ICER '20, page 194–203, New York, NY, USA, 2020. Association for Computing Machinery. → pages 11
- [MKKN05] Lauri Malmi, Ville Karavirta, Ari Korhonen, and Jussi Nikander. Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. *J. Educ. Resour. Comput.*, 5(3):7–es, sep 2005. → pages 11
- [MNS⁺20] Hamza Manzoor, Amit Naik, Clifford A. Shaffer, Chris North, and Stephen H. Edwards. Auto-grading jupyter notebooks. In Jian Zhang, Mark Sherriff, Sarah Heckman, Pamela A. Cutter, and Alvaro E. Monge, editors, *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE 2020, Portland, OR, USA, March 11-14, 2020*, pages 1139–1144. ACM, 2020. → pages 7
- [MSR21] Mostafa Mohammed, Clifford A. Shaffer, and Susan H. Rodger. Teaching formal languages with visualizations and auto-graded exercises. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, pages 569–575. ACM, 2021. → pages 7
- [Pie13] Vreda Pieterse. Automated assessment of programming assignments. In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*, CSERC '13, page 45–56, Heerlen, NLD, 2013. Open Universiteit, Heerlen. → pages 12
- [SBP⁺10] Josep Soler, Imma Boada, Ferran Prados, Jordi Poch, and Ramón Fabregat. A formative assessment tool for conceptual database design using UML class diagram. *iJET*, 5(3):27–33, 2010. → pages 8
- [Sch20] Johannes Schildgen. Monster park - the entity-relationship-diagram learning game. In *ER Forum, Demo and Posters 2020 co-located with 39th International Conference on Conceptual Modeling (ER 2020), Vienna, Austria, November 3-6, 2020*, volume 2716 of *CEUR Workshop Proceedings*, pages 150–157. CEUR-WS.org, 2020. → pages 9

- [STW13] Neil Smith, Pete G. Thomas, and Kevin G. Waugh. Automatic grading of free-form diagrams with label hypernymy. In *2013 Learning and Teaching in Computing and Engineering, LaT-iCE 2013, Macau, Macao, March 21-24, 2013*, pages 136–142. IEEE Computer Society, 2013. → pages 7
- [SvdPSC19] Dave R. Stikkolorum, Peter van der Putten, Caroline Sperandio, and Michel Chaudron. Towards automated grading of UML class diagrams with machine learning. In *Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019)*, volume 2491 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019. → pages 7
- [TPE05] Scott A. Turner, Manuel A. Pérez-Quiñones, and Stephen H. Edwards. minimuml: A minimalist approach to UML diagramming for early computer science education. *ACM J. Educ. Resour. Comput.*, 5(4):1:1–1:28, 2005. → pages 6
- [TWS06] Pete G. Thomas, Kevin G. Waugh, and Neil Smith. Using patterns in the automatic marking of er-diagrams. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2006, Bologna, Italy, June 26-28, 2006*, pages 83–87. ACM, 2006. → pages 1, 7
- [WHZ15] Matthew West, Geoffrey L. Herman, and C. Zilles. Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. In *122nd ASEE Annual Conference and Exposition*, pages 26.1238.1 – 26.1238.14, 2015. → pages 1, 10, 14
- [Zak04] Konstantin Zakharov. Feedback micro-engineering in eer-tutor. 2004. → pages 6

Appendix

Appendix A

Consent Forms

A.1 Administrator Consent



THE UNIVERSITY OF BRITISH COLUMBIA

Project Title: AutoER - ER/UML Diagram Designer and Evaluator
Human Ethics - H21-01600
Principal Investigator: Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca
Department of Computer Science, Physics, Mathematics, Statistics UBCO
Co-Investigator: Sarah Foss
Email: sarafoss@mail.ubc.ca

INFORMED CONSENT: COURSE ADMINISTRATOR PARTICIPATION

Introduction

You are invited to participate in an evaluation of an experimental educational tool for generating and assessing ER diagrams. As a participant in this study, you will be asked to complete a questionnaire and interact with this tool as needed.

This statement contains information about the present study that is intended to help you decide whether you wish to participate in it. You may refuse to sign this form and not participate in this study. You should be aware that even after you signed, you are free to withdraw at any time. If you withdraw from this study, it will not affect your relationship with this unit, the people involved in the study, the services it may provide to you, or the University of British Columbia Okanagan. If you have any questions about this study before, during, or after your participation, please feel free to direct them to:

Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca

Study Objectives

The objective of this work is to explore the utility of allowing students to interact with entity-relationship diagram questions in a visual way that will allow the system to generate a diagram that can be assessed automatically and consistently.

Procedure: ER Diagram Designer and Evaluator

Total expected time: 3 hours

1. The co-investigator will visit your class to explain the study and the consent form to the students.
2. For the assignments and exams that require students to draw ER diagrams, you will log into the tool and enter the questions for the assignment/exam and the answers in the required format.
3. As part of the course, your students will work on questions related to ER diagrams in the software directly.
4. After their work has been submitted, you will retrieve their answer and manually grade the student's work as usual. If necessary, you may override the feedback or grades generated by the system.
5. At the end of the course, you will be asked to complete a short questionnaire based on the utility and usability of the tool. This is to help us identify what has been useful for you and what changes we need to make in the future.

Risks

There is a risk that the software will malfunction and negatively impact or delay student's grades. To mitigate this, the course administrator will manually mark each submission and, if necessary, they will have the ability to override marks and add feedback for the students in the software

Benefits

The student participants may gain a better understanding of the concepts of entity-relationship diagrams through immediate visual representation. They will also benefit from consistent marking from the software.

The course administrators will benefit from the time-reducing auto marking capabilities of the software.

Privacy and confidentiality

Questionnaires are anonymous. Questionnaire data, consent data, and information collected from the system will all be encrypted and stored separately. Since we are only interested in average responses and behaviors, reports about individual students will not be released.

Participants have a right to access their own results as long as their names and numeric tags are still available. The data obtained from this study will be retained for at least 5 years after publication in secured, electronic form, which only researchers associated with this research will have access to. No audio or video tapes will be used. General results of the research study may be available through publications. A summary of the results will be made available to participants upon request.

Concerns about participants' rights

If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Ethics toll free at 1-877-822-8598 or the UBC Okanagan Research Services Office at 250-807-8832. It is also possible to contact the Research Complaint Line by email (RSIL@ors.ubc.ca). When doing so, include the study number H21-01600 when contacting their staff so they can better assist you.

Participant Certification

I have read this Consent and Authorization form. I have had the opportunity to ask, and I have received answers to, any questions I had regarding the study and the use and disclosure of information about me for the study. I agree to take part in this study as a research participant. By my signature I affirm that I have received a copy of this Consent form.

Participant Name:

Date:

Participant Signature:

A.2 Student Consent



THE UNIVERSITY OF BRITISH COLUMBIA

Project Title: AutoER - ER/UML Diagram Designer and Evaluator
Human Ethics - H21-01600
Principal Investigator: Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca
Department of Computer Science, Physics, Mathematics, Statistics UBCO
Co-Investigator: Sarah Foss
Email: sarafoss@mail.ubc.ca

INFORMED CONSENT: STUDENT PARTICIPATION

Introduction

You are invited to participate in an evaluation of an experimental educational tool for generating and assessing ER diagrams. As a participant in this study, you will be asked to complete a questionnaire and interact with this tool as needed.

This statement contains information about the present study that is intended to help you decide whether you wish to participate in it. You may refuse to sign this form and not participate in this study. You should be aware that even after you signed, you are free to withdraw at any time. If you withdraw from this study, it will not affect your relationship with this unit, the people involved in the study, the services it may provide to you, or the University of British Columbia Okanagan. If you have any questions about this study before, during, or after your participation, please feel free to direct them to:

Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca

Study Objectives

The objective of this work is to explore the utility of allowing students to interact with entity-relationship diagram questions in a visual way that will allow the system to generate a diagram that can be assessed automatically and consistently.

Procedure: ER Diagram Designer and Evaluator

Expected course work time: 4 hours

Expected additional time: 0.5 hours

Total expected time: 4.5 hours

1. As part of the course, you will work on questions related to ER diagrams in the software directly.
2. After your work has been submitted and automatically graded, your instructor will also manually grade your answers to verify they were marked correctly. If necessary, they are able to override the feedback or grades.
3. At the end of the course, you will be asked to complete a short questionnaire based on the utility and usability of the tool. This is to help us identify what has been useful for you and what changes we need to make in the future.

Risks

There is a risk of the software malfunctioning and negatively impacting your grades in the course. To mitigate this, the course administrator will also manually mark each submission and, if necessary, they will have the ability to override marks and add feedback in the software.

There is a risk that you may become frustrated with the software. To mitigate this, you will have the option of using traditional methods of creating entity-relationship diagrams at any time.

Version: 1.1
June 29, 2021

1

Benefits

You may gain a better understanding of the concepts of entity-relationship diagrams through immediate visual representation. You will also benefit from consistent marking from the software.

Privacy and confidentiality

Questionnaires are anonymous. Questionnaire data, consent data, and information collected from the system will all be stored separately and encrypted. Since the interest of this study is to identify average responses and behavior of the entire group of participants, you will not be identified in any way in any written reports of this research.

Participants have a right to access their own results so long as their names and numeric tags are still available. The data obtained from both components of the study will be retained for at least 5 years after publication in secured, electronic form, which only researchers associated with this research will have access to. No audio or video tapes will be used. General results of the research study may be available through publications. A summary of the results will be made available to participants upon request.

Access to course work and associated grades

By checking the boxes below, you are providing consent for the researchers in this study to analyze your entity-relationship diagram course work and associated grades. The grades will only be used for research analysis purposes without including any of the participants' personal information.

Concerns about participants' rights

If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Ethics toll free at 1-877-822-8598 or the UBC Okanagan Research Services Office at 250-807-8832. It is also possible to contact the Research Complaint Line by email (RSIL@ors.ubc.ca). When doing so, include the study number H21-01600 when contacting their staff so they can better assist you.

Participant consent and signature

I have read this Consent form. I have had the opportunity to ask, and I have received answers to, any questions I had regarding the study and the use and disclosure of information about me for the study. I agree to take part in this study as a research participant. By my signature I affirm that I have received a copy of this Consent form.

<input type="checkbox"/> Yes <input type="checkbox"/> No	I am willing to provide access to my submitted entity-relationship diagram course work
<input type="checkbox"/> Yes <input type="checkbox"/> No	I am willing to provide access to my course grades associated with the submitted work.

Whether or not you choose to participate, you will receive the same assignments and exam questions in this course. Your decision to participate or not participate in this study will in no way impact your grades or your relationship with UBC.

Participant Name:

Date:

Participant Signature:

Appendix B

Surveys

B.1 Student Survey



THE UNIVERSITY OF BRITISH COLUMBIA

Project Title: AutoER - ER/UML Diagram Designer and Evaluator
Human Ethics - H21-01600
Principal Investigator: Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca
Department of Computer Science, Physics, Mathematics, Statistics UBCO
Co-Investigator: Sarah Foss
Email: sarafoss@mail.ubc.ca

Participant ID: _____ Date: _____

Questionnaire - AutoER – ER/UML Diagram Designer and Evaluator Software

Thank you for your voluntary participation in the AutoER - ER/UML Diagram Designer and Evaluator research study under the direction of Dr. Ramon Lawrence.

In order to improve the software used in this study, this questionnaire has been developed to gather feedback regarding your experiences using the software. We value your honest and detailed responses. Your responses are confidential.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would like to use this tool frequently.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. I found the tool unnecessarily complex.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the tool was easy to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this tool were well integrated.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this tool very quickly.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the tool very cumbersome to use.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this tool.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

11. Select all that apply. I used the software for the following tasks:

- At least one lab
- More than one lab
- Midterm exam
- Final exam
- Did not use the software

12. If applicable, why did you choose to keep using the software?

13. If applicable, why did you choose to stop using the software?

14. What aspects of the software did you find helpful?

15. What additional features would you like to see with this software?

16. Please provide any general comments about the software and its use in the course.

B.2 Administrator Survey



THE UNIVERSITY OF BRITISH COLUMBIA

Project Title: AutoER - ER/UML Diagram Designer and Evaluator
Human Ethics - H21-01600
Principal Investigator: Dr. Ramon Lawrence
Email: ramon.lawrence@ubc.ca
Department of Computer Science, Physics, Mathematics, Statistics UBCO
Co-Investigator: Sarah Foss
Email: sarafoss@mail.ubc.ca

Participant ID: _____ Date: _____

Questionnaire - AutoER – ER/UML Diagram Designer and Evaluator Software

Thank you for your voluntary participation in the AutoER - ER/UML Diagram Designer and Evaluator research study under the direction of Dr. Ramon Lawrence.

In order to improve the software used in this study, this questionnaire has been developed to gather feedback regarding your experiences using the software. We value your honest and detailed responses. Your responses are confidential.

1. Select all that apply. I used the software for the following tasks:

- At least one lab
- More than one lab
- Midterm exam
- Final exam
- Did not use the software

2. Select one. I found the software easy to use.

- Strongly agree
- Somewhat agree
- Neither agree nor disagree
- Somewhat disagree
- Strongly disagree

3. Select one. I found the software useful.

- Strongly agree
 - Somewhat agree
 - Neither agree nor disagree
 - Somewhat disagree
 - Strongly disagree
4. Select one. The software reduced the time spent grading ER diagrams.
- Strongly agree
 - Somewhat agree
 - Neither agree nor disagree
 - Somewhat disagree
 - Strongly disagree
5. If applicable, why did you choose to keep using the software?
6. If applicable, why did you choose to stop using the software?
7. What aspects of the software did you find helpful?
8. What additional features would you like to see with this software?
9. Do you have an estimate on how much time the software saved for grading?
10. Please provide any general comments about the software and its use in the course.

Appendix C

Assignments

C.1 Assignment 1

Building ER diagrams using an UML modeling tool

This lab designs ER diagrams in UML notation. Students participating in the AutoER study can use the AutoER Designer and Evaluator software that performs real-time marking and feedback.

It is also possible to use a variety of drawing software such as astah. Note: Astah no longer has a free community version. Request a student license or use the 30-day trial. diagrams.net is an online drawing tool that can also be used but it does not have as good support for database modeling in UML.

C.1.1 Question 1 (10 marks)

Construct a database design in UML for an Online Game store described below.

A **developer** where each **developer** is identified by an **id** and has a **name**.

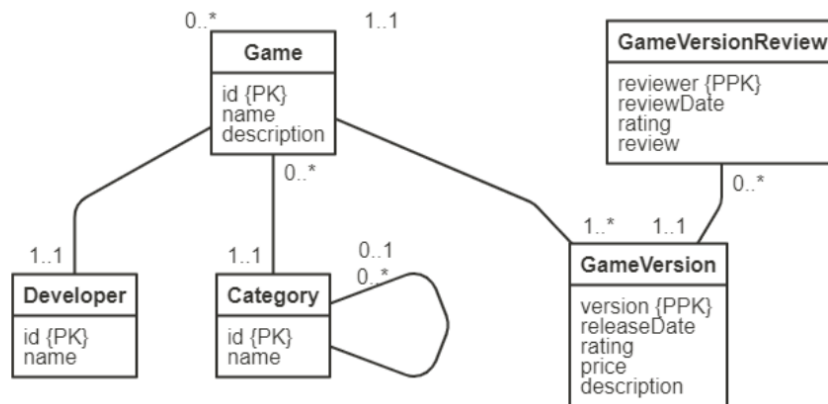
A **category** where each **category** has an **id**, a **name**, and may have a parent **category**.

A **game** storing each **game** that is identified by a field called **id** and other attributes include **name** and **description**. A **game** is created by one **developer**. A **developer** may publish multiple **games**. A **game** has a **category**.

A **game version** stores each version of the game. A **game version** is associated with exactly one **game**. Use a **version** field to identify between **versions** of the same **game**. Each **game version** has a **release date**, a **rating**, a **price**, and a **description**.

A **game version review** stores **ratings** for each game version. Each **instance** applies to a single **game version**, and different **reviews** are identified by **reviewer** attribute (which is name of reviewer). There is also a **review date**, **rating**, and **review**.

(a) Question



(b) Answer

Figure C.1: Assignment 1 Question 1

C.1.2 Question 2 (10 marks)

There are multiple **schools** in the school management system. A **school** is identified by its **name** and has a **location**.

A **teacher** is identified by their **teacher number** and has a **name**. Each **school** has a single **teacher** as a principal, and a **teacher** may be the principal at only one **school**.

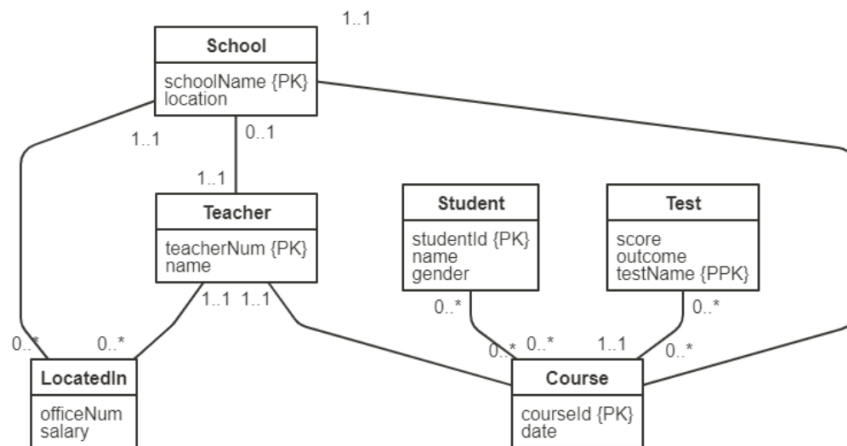
Teachers are **located in schools**. A teacher may be **located in** more than one **school**. A teacher located at a school has an **office number** and a **salary** paid by that **school**.

A **student** is identified by their **student id** and also has a **name** and **gender**.

A **student** takes a **course** with a **teacher** at a particular **school**. Each **course** is identified by its own **id** and also has a **meeting date**.

At a **course** zero or more **tests** are run each with a **score** and an **outcome {Pass/Fail}**. Each **test** is identified for a particular **course** by **name**.

(a) Question



(b) Answer

Figure C.2: Assignment 1 Question 2

C.2 Assignment 2

C.2.1 Question 1 (15 marks)

Note: In addition to the diagram, submit a document containing the mapping to the relational model. SQL CREATE TABLE statements are not required. Just provide relation names, attributes, keys and foreign keys.

C.2. Assignment 2

Draw the ER diagram for tracking football statistics for quarterbacks and runningbacks in college football:

The league will have multiple **teams**, each with a unique **team name**.

For each **game** played, there is a **home team**, an **away team**, **home points**, **away points**, and a **date**.

A **game** is identified by the **home team**, an **away team**, and the **date**.

All **teams** play multiple home and away games per season.

The teams all have **quarterbacks** and **runningbacks** that are identified by **team name** and **jersey number**. Also store a player **name**.

The **team name** and **number** will be unique for each player, while their **name** may not be unique.

Each **runningback** has a **type** {fullback or halfback}. Each **quarterback** has a **status** {starter, backup}.

Statistics are compiled for each **game** for each player.

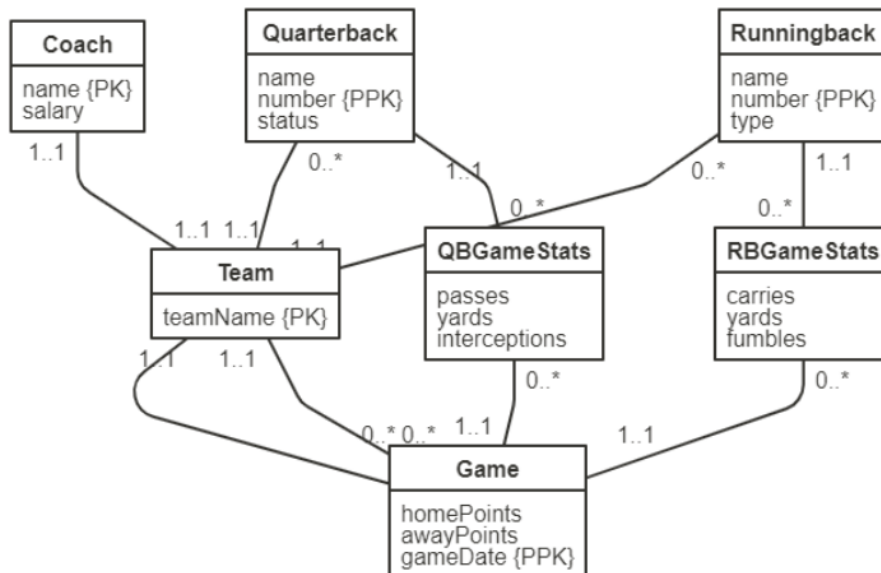
The **runningback statistics** will include **carries**, **yards**, and **fumbles**.

The **quarterback statistics** include **passes**, **yards**, and **interceptions**.

In addition, each team is represented by a single **coach**.

A **coach** can coach only one **team**. Keep track of each coach's **name** and **salary**.

(a) Question



(b) Answer

Figure C.3: Assignment 2 Question 1

C.2.2 Question 2 (35 marks) - Project Deliverable

The project will build an online store like Amazon.com selling whatever products you want. The first step is to develop a database design and convert that design into SQL Data Definition Language (DDL).

Deliverables:

1. Draw the ER/UML diagram for this database. (21 marks)
2. Convert the UML diagram into SQL DDL. Make sure to define primary keys and foreign keys. Your SQL DDL must run on either MySQL, Microsoft SQL Server or Oracle APEX (preferable). (11 marks)
3. Determine what you are going to sell in your store. Write a mission statement (1 mark) and executive summary paragraph (2 marks) describing your idea.

Items to submit: your UML diagram, a text file containing your SQL DDL, a document with your mission statement and executive summary.

C.2. Assignment 2

Draw the ER diagram for tracking football statistics for quarterbacks and runningbacks in college football:

The league will have multiple **teams**, each with a unique **team name**.

For each **game** played, there is a **home team**, an **away team**, **home points**, **away points**, and a **date**.

A **game** is identified by the **home team**, an **away team**, and the **date**.

All **teams** play multiple home and away games per season.

The teams all have **quarterbacks** and **runningbacks** that are identified by **team name** and **jersey number**. Also store a player **name**.

The **team name** and **number** will be unique for each player, while their **name** may not be unique.

Each **runningback** has a **type** {fullback or halfback}. Each **quarterback** has a **status** {starter, backup}.

Statistics are compiled for each **game** for each player.

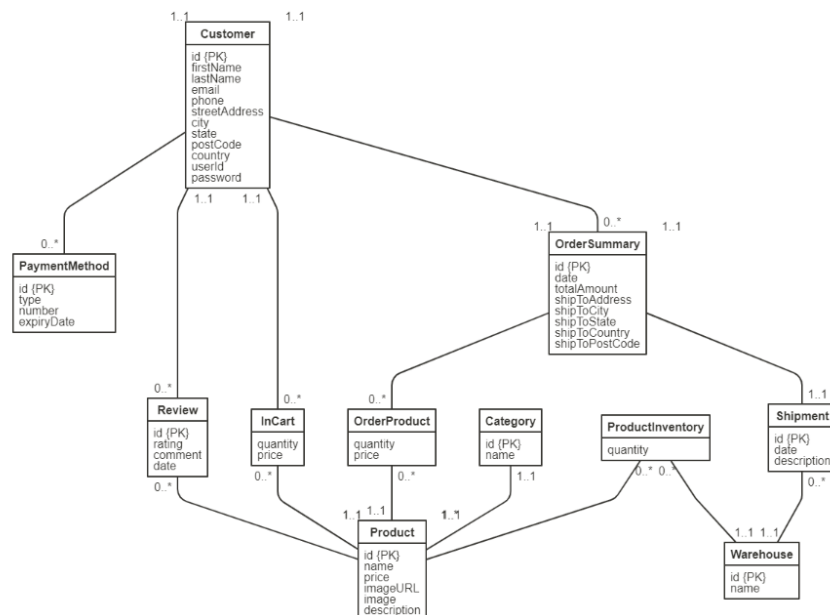
The **runningback statistics** will include **carries**, **yards**, and **fumbles**.

The **quarterback statistics** include **passes**, **yards**, and **interceptions**.

In addition, each team is represented by a single **coach**.

A **coach** can coach only one **team**. Keep track of each coach's **name** and **salary**.

(a) Question



(b) Answer

Figure C.4: Assignment 2 Question 2