AUTOMATED GRADING OF UML USE CASE DIAGRAMS

A Thesis Submitted to the Committee on Graduate Studies

in Partial Fulfillment of the Requirements for the degree of Master of Science

in the Faculty of Arts and Science

PREVIEW

TRENT UNIVERSITY

Peterborough, Ontario, Canada

© Copyright By Mohsen Hosseinibaghdadabadi 2023

Applied Modelling & Quantitative Methods M.Sc. Graduate Program

May 2023

ABSTRACT

Automated Grading of UML Use Case Diagrams
Mohsen Hosseinibaghdadabadi

This thesis presents an approach for automated grading of UML Use Case diagrams. Many software engineering courses require students to learn how to model the behavioural features of a problem domain or an object-oriented design in the form of a use case diagram. Because assessing UML assignments is a time-consuming and labor-intensive operation, there is a need for an automated grading strategy that may help instructors by speeding up the grading process while also maintaining uniformity and fairness in large classrooms.

The effectiveness of this automated grading approach was assessed by applying it to two real-world assignments. We demonstrate how the result is similar to manual grading, which was less than 7% on average; and when we applied some strategies, such as configuring settings and using multiple solutions, the average differences were even lower. Also, the grading methods and the tool are proposed and empirically validated.

PREVIEW

Keywords: automated grading, use case diagram, model comparison

Acknowledgment

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Omar Alam, and also Dr. Jorg Kienzle from McGill University for the continuous support of my master's study and research, for your patience, motivation, enthusiasm, and immense knowledge. Your guidance helped me in all the time of research and writing of this thesis. Also, thank you to my supervisor committee for your time and advice.

My heartfelt appreciation goes to my wife, Atifeh, who endured all the hardships of being away from her family, friends, and homeland, she was by my side throughout this whole process and gave me motivation and hope whenever I was disappointed so that I could achieve my lofty goals.

I am also eternally grateful to my parents and siblings. They preferred my success to being away from them and motivated me all along the way to continue with strength until the end.

I owe a great deal of gratitude to my University, Trent, and all its employees for providing me with such a valuable opportunity to study for my Master's in my favorite field.

Finally, I am grateful to my family and friends for their unwavering love and support, as well as for instilling in me the confidence and determination to attain my lofty ambitions. I also thank my dear friend Bahadur for all his help in this way. While I cannot expect to adequately thank everyone who has helped me with my MSc, I would like to offer my heartfelt gratitude to everyone for their help along this great journey.

PREVIEW

Contents

PREVIEW

PREVIEW

PREVIEW

List of Figures

PREVIEW

PREVIEW

PREVIEW

List of Tables

PREVIEW

Chapter 1

Introduction

1.1 Motivation

Engineers use models to explain the shapes or actions of any structure they wish to build or develop. Typically, graphic symbols, relations, and explanations, will be designed to visualize the model[1]. As use case diagram is kind of a behavior diagram in the Unified Modeling Language[2], it is used to represent the dynamic behavior of a system and understand how the system should work [3].

The growth of new technology requires considerable adjustments in the educational system [4]. The advancement of computer-based information technology can facilitate learning, particularly in educational assessments or evaluating student learning outcomes [5, 6]. Due to the growing number of computer science students, one of the most important procedures in teaching and learning is assessment [7, 8]. When assessing each assignment, evaluating the student use case model might be challenging for the teacher [9]. A teacher must conduct an assessment method while teaching use case diagrams in this scenario. Based on the answer keys, a teacher will grade the use case diagrams created by students. This use case diagram can be very long, including many use cases, and since each use case contains a specification part comprising many texts, grading each assignment will take considerable energy and time. Furthermore, courses with a large number of participants cannot provide individual support in solving modelling tasks, as there are often multiple correct solutions. The use case diagram created by students may differ from the instructor's solution due to using different words when

PREVIEW

naming actors and use cases, or paraphrasing the use case Main Success Scenario steps. However, the student's answer may differ from the instructor's, but the whole process can be very similar, which causes difficulties for the instructor when grading a number of students' solutions manually. Moreover, individual feedback on the modelling solution is extremely crucial in this situation [9]. In addition to the increasing workload, instructors struggle to fairly mark assignments and tests, which is a difficult process, particularly, when matching texts and flattening steps become important in the use case specification part. Furthermore, teachers frequently alter their marking scheme after assessing multiple student papers. For example, instructors may decide to redistribute scores if they see that students struggled with a certain area of the model, indicating that the issue description was possibly confusing. To compensate, the instructor may decide to modify the grading weights for different parts of the model.

Besides, at a time when the internet has reached almost everywhere, automated grading is critical for e-learning and Massive Open Online Courses (MOOCs) [10, 11]. These online courses offer a very accessible means for learners to develop new and existing skills. Self-assessment methodologies have been integrated into popular online learning systems such as Linkedin [12], Coursera [13], EDX [14], and Skillshare [15]. Automated assessment can also be used to calibrate a learner's past knowledge [16], that is, to test the learner's prior knowledge of the subject before starting a new course. Furthermore, in the context of online courses, automated grading may be employed to offer learners fast feedback [17].

Motivated by the reasons stated above, we propose an automated grading approach for UML use case diagrams. We reused metamodels introduced by Bian. et al [18]. These grading metamodels were implemented in the TouchCORE tool [19], which visually displays the marks on the classes and prints feedback to the student.

1.2 Problem Summary

This thesis is an attempt to address the issue highlighted in the introduction. First, we want to see if automated grading is more effective for use case models than manual grading. As a result, by reusing metamodels introduced by Bian et

PREVIEW

al. [18], we propose an automated grading approach for UML use case diagrams . these grading metamodels were implemented in the TouchCORE tool [19], which visually displays the marks and prints feedback to the student. We offer an algorithm that uses syntactic, semantic, and structural matching techniques to create mappings between model components in the instructor's solutions and elements in the student solutions. We also demonstrate how flattening steps improves the grades of students and decreases instructor mistakes when grading a large use case assignment. In addition, our algorithm can compare those flattened steps, conceptually using some natural language processing techniques.

We ran this algorithm on real assignments for modeling a Gas Station for a class of 14 undergraduate students, and Elfenroads use case models for 9 undergraduate student groups. On average, our algorithm was able to automatically grade the assignments within a 7% difference from the instructor's grade.

1.3 Thesis Contributions

We offer an algorithm that uses syntactic, semantic, structural, and text matching techniques to create mappings between model components in the instructor's solutions and elements in the student solutions. Each precisely matched element earns the students a full mark. Partially right answers, such as a correct actor with incorrect multiplicity, receive a partial mark and earn no marks if the component is missed or completely incorrect. One significant advantage of our technique is that it can easily update the students' marks when the instructor changes the grading scheme. Additionally, we propose a flattening steps approach, which orders all steps in the instructor's and student's solution into a list, and since those steps play an important role in the use case model, our algorithm can compare them conceptually using some natural language processing techniques.

We evaluated our algorithm using modelling two real-world assignments; a Gas Station and Elfenroads game. At this point, there are 14 individual submissions for one case study and nine group submissions of seven students for another, which means 63 students contributed to this assignment. Although our automated grading results are close to manual grading and provide confident evidence, our

PREVIEW

sample size might not be large enough, and we need to examine more case studies in the future.

1.4 Research Questions

The following research questions can be answered by comparing the outcomes of automated grading with manual grading performed by instructors:

- **RQ1: How different are manual and automated grading?**

  There should not be a large difference between an instructor's grades and automated grades. Using two case studies, we will demonstrate that the average difference is less than 9% when using default settings.

- **RQ2: Does the use of configuration settings improve the accuracy of automated grading?**

  A configurable grading algorithm can generate results that are more similar to the instructor's manual grading scores. Overall, the difference in one case study decreased from 9.59% to 6.69%, while the difference in another case study is reduced from 8.27% to 7.61%.

- **RQ3: Does automated grading help ensure fairness?**

  Inconsistencies are common with manual grading, and we investigated the consistency aspect of fairness. Automated grading can detect discrepancies in manual grading. We will explain how the instructor could make mistakes when grading.

- **RQ4: Does the accuracy of automated grading improve when multiple solutions are matched against?**

  There could be more than one correct solution for a use case diagram modelling problem. So, the tool is capable of loading multiple correct solutions. When all three available solutions were used in one case study, the average differences decreased from 6.69% to 4.50%.

- **RQ5: How do flattening steps helps automated grading to achieve a better result?**

PREVIEW

When there are multiple use cases in the diagram, they may be linked by a relationship. The best idea when grading is to first flatten all steps in all use cases based on their relationships. We proceed with this method from the root use case to the leaf use cases in the diagram, and at the end, we have a list of flattened steps. When manually grading large models, flattening can be a difficult task. When grading without flattening is compared to grading with flattening, it can be seen that the average mark is significantly improved in both case studies.

- **RQ6: Does using NLP improve the quality of the grading algorithm?**

  The most challenging part of automated grading in use case models is matching texts. The matching text algorithm is essential in the grading of use case models. The algorithm is more effective when we combine it with some natural language processing methods. The thesis examines and shows how to find the best threshold value that can be set in the algorithm when matching texts. In addition, different parts of speech could have different levels of importance. Hence, coefficient weights for two kinds of parts of speech (nouns and verbs) are examined, and as a result, we should consider weights for nouns and verbs 1.5 times more than other parts of speech.

PREVIEW

1.5 Thesis Organization

This thesis is divided into seven chapters. In Chapter 2, we address related works on automated grading. In Chapter 3, we then present examples that motivate our matching strategies. Chapter 4 describes several methodologies for matching words and texts and also discusses the algorithms that compare the student's model to the instructor's model. The architecture, metamodels, and grading tools that facilitate automated grading are demonstrated in Chapter 5. In this chapter, we also review our configuration panel and elaborate on the configuration options for automated grading. The case study arrangement is described in detail in Chapter 6. We also evaluate our tool and answer research questions by comparing results in this chapter. Chapter 7 concludes this thesis and discusses directions

for future work.

The contributions of the thesis are organized into five parts:

Part 1: Motivation (Chapter 3)

We motivate our approach using a use case diagram modelling the "Elfenroads" game. All the strategies we need to match and grade each component of the diagram are discussed in this part.

Part 2: Grading Algorithm (Chapter 4)

The thesis proposes and describes various matching techniques for grading the UML use case diagram. Motivating model examples are used to demonstrate the process. It also goes over the methodologies used in matching words and texts, as well as demonstrating the algorithm for automated grading of UML use case diagrams.

Part 3: Grading Architecture and Tool Support (Chapter 5)

The architecture of the grading strategy and two metamodels are introduced in the thesis to enable the automated grading algorithm. Furthermore, the thesis describes the grading tool that implements the automated grading algorithm.

Part 4: Case Study (Chapter 6)

The grading technique is used in the thesis for a case study of two modeling tasks.

Part 5: Research Questions (Chapter 6)

In this section, six research questions mentioned in Section 1.4 will be investigated in detail.

PREVIEW

Chapter 2

Related Work

2.1 Automated Grading Tools

Automated grading approaches has been explored for other modeling languages. Alur et al. [20] proposed a method for automatically grading the standard computation-theory problem, which asks students to construct a deterministic finite automaton (DFA) from a description of its language. They focused on how to assign partial grades for incorrect students' answers using a hybrid of three techniques: syntactic, semantic, and capturing mistakes. Batmaz and Hinde [21] presented a system to assist human graders in evaluating conceptual database diagrams. It also helps students model their diagrams easily. They focused on semi-automatic diagram marking, aiming to reduce the number of sub-diagrams marked by the examiner. Simanjuntak [22] presented preliminary research and a proposed framework for an automated ER Diagram grading system. He presented two approaches: The Tree Edit Distance algorithm to measure ER Diagram similarity, and a machine learning technique to build a classifier that automatically grades ER Diagrams. Thomas et al. [23] provided a method for computer-aided diagram understanding and demonstrated how it may be effectively used for the automatic grading of student attempts at drawing entity-relationship (ER) diagrams. They concentrated on imprecise diagrams, in which the required characteristics are either malformed or missing, or extraneous features are included.

There are approaches that propose the automated assessment of other kinds of UML models. Bian et al. [18] presented an approach for automated grading of

PREVIEW

UML class diagrams. They proposed a metamodel that establishes mappings between the instructor's solution and all the solutions for a class. Their approach employs a grading algorithm that uses syntactic, semantic, and structural matching to match a student's solutions with the template solution. They also investigated the effectiveness of an automated grading approach for UML class diagrams when applied to two classroom settings across different universities and compared the results of automated grading with manual grading. [24]. Striewe et al. [25] describe a method for grading UML behavioural diagrams based on trace information. Tsintisfas [26] created a generic student diagram editor that can be easily customised for the exercise. He also produced a Computer Based Assessment (CBA) that automatically marks diagram-based models, e.g., process and entity-relationship diagrams. Jayal and Shepperd [27] developed a label-matching approach for UML diagrams that employs multiple levels of decomposition as well as syntactical matching. Through text transformation processing, they empirically investigated the diversity of labels used by students in the activity diagram. Thomas et al. [28] presented the analysis and classification of errors in imprecise sequence diagrams as well as the performance of an automated grading system. The analysis informed the design of a syntax error-checking tool that detects and reports on syntax errors and also repairs the majority of error types using information gleaned from the error analysis. Tselonis et al. [29] proposed a graph-matching-based diagram marking approach for structural diagrams, e.g., UML class diagrams and electronic circuits. Sousa and Paulo [30] proposed a structural technique for graphs that develops mappings from an instructor's solution to elements in a student's solution in order to maximize a student's grade. The proposed algorithm is applicable to any type of document that can be parsed into its graph-inspired data model. This data model is able to accommodate diagram languages, such as UML or ER diagrams, for which this kind of assessment is typically used.

PREVIEW

2.2 Automated Grading Tools for UML Use Case Diagrams

Limited work has been done in the area of automated evaluation of use case diagrams, or it is under process. Fauzan et al. [3] created a semantic use case diagram for an automatic assessment method that performs WordNet [31] searches using WuPalmer [32, 33]. They first translated the use case diagram's features (actor, use case, relationships) into XMI and matched the labels semantically using some natural language processing methods. Vachharajani et al. [34] proposed an architecture of automated assessment of use case diagrams. The essence of their architecture is to assess a large number of students' diagrammatic answers very easily in a short period of time. Their tool can also give quantitative feedback in terms of grades as well as qualitative feedback in terms of suggestions. Vachharajani et al. [35] also employed a hybrid technique for automatically assessing use case diagrams, in which labels were processed simultaneously in the syntactic and semantic matching phases. They considered a use case diagram as a mixed graph, with nodes representing different actors or use cases and edges indicating the various relationships among them. In another paper, Vachharajani et al. [36] introduced and developed a tool named "Use Case Extractor" towards achieving the research objective of automated evaluation of the Use Case diagram. Arifin et al. [37] proposed a method for measuring the similarity of use case diagrams using structural and semantic matching aspects. Their proposed method used the process of modelling the use case diagram as a graph and the graph similarity method to measure structural similarity, and WuPalmer [32] and Levenshtein [38] to measure semantic similarity. Kumar et al. [39] introduced the Static UML Model Generator from Analysis of Requirements (SUGAR) tool, which generates both use case and class models while emphasizing natural language requirements. SUGAR aims to bring together both the requirement analysis and design phases by identifying use cases, actors, classes, attributes, and methods, as well as proper class association. They discussed the tool that generates all static UML models in Java in conjunction with Rational Rose.

PREVIEW

2.3 Conclusion

There are some significant differences between our method and the aforementioned approaches. The majority of them focused their research on grading the diagram, name (label), and relationship matching rather than the use case specification. They used syntactic and semantic matching to compare names and rarely used NLP methods for texts with more than one word. They also considered structural matching but could not match a use case with the wrong name based on its specification. We also couldn't find a work that matched the text (step) or took flattening steps into account.

On the other hand, our approach uses syntactic and semantic matching as well as NLP methods to match names and texts, e.g., use case multiplicity and steps. We used Stanford CoreNLP to tokenize, lemmatize, recognize names of entities, find parts of speech, and remove unimportant words before matching texts. We also consider the order of steps and match orders in addition to texts. Using use case specifications, we could figure out split use cases and determine which student used the wrong name based on the steps of that use case. So, the use case is correct, and the specification matches, but the student used the wrong name. The method that distinguishes our approach from others' works is flattening. When flattening all use case steps, all steps in the model will be compared, and students will receive points for any correct steps, regardless of whether they have used the correct name for the use case. Furthermore, more than one correct solution can be uploaded, and the student's solution compares with all possible solutions. The instructor is also able to assign points or redesign the marking scheme for each component and feature in the model and can manage the deduction percentage through the configuration settings. Finally, the tool provides the student and instructor with detailed feedback, indicating where the student lost points.

PREVIEW

Chapter 3

Motivation

In this chapter, we will motivate our approach using a use case diagram modelling the "Elfenroads" game. The method proposed here will be applied in our automated grading system, which will be discussed in depth in Chapter 4.

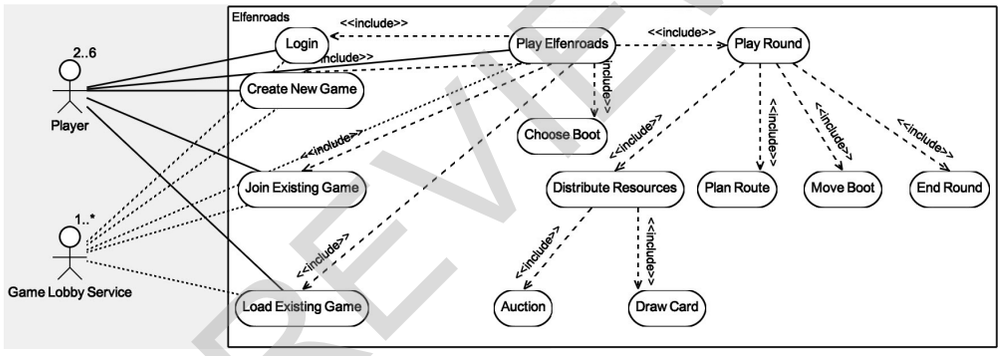

Figure 3.1: Instructor Solution for Elfenroads Use Case Model

The instructor's use case diagram solution, depicted in Fig. 3.1, contains 13 use cases. Play Elfenroads, Login, Create New Game, Join Existing Game, and Load Existing Game are all use cases that are linked to the "Player" as the primary actor and the "Game Lobby Service" as the secondary actor. One student's solution, shown in Fig. 3.2, uses "Aoction" which is the wrong spelling form for "Auction". This student also uses the word "Connect Game" as a name for the use case "Join Game", and uses "Create Game" instead of "Create New Game". Although the words that were used are not the same, we want our matching algorithm to determine that "Connect Game" is a synonym for "Join Game", which is called a semantic match. The use case "Aoction" should be matched with "Auction"

PREVIEW

Reproduced with permission of copyright owner. Further reproduction prohibited without permission.