Automated Review Tool for Educational Models Utilizing Generative AI

Daisuke Saito[†1] Taiyou Minagawa[†1] Kenji Hisazumi [†1]

Keywords: UML Modeling, Class Diagram, Review, Generative AI, ChatGPT

1. Introduction

Improving design quality has become increasingly significant in recent system development endeavors. Furthermore, design through modeling is indispensable in system development, and the quality of a model plays a pivotal role in determining the quality of the design. Consequently, engineers are now faced with the demand to enhance their modeling techniques for design, underscoring the growing importance of education in modeling.

In the realm of modeling education, there is a vast array of ways to create a model. To make education more effective, it is essential for students to receive feedback from their instructors. However, there's no singular *correct* model, and the means of expression can vary from one to another. Thus, the task of individually reviewing and grading each model, then providing tailored feedback, imposes a significant burden on instructors and is time-consuming.

This research proposes a tool to support modeling education by automatically grading models written by students and aiding the feedback process of instructors. Our proposed method focuses on frequently used class diagrams. It auto-grades student-created models and offers feedback on errors, laying the groundwork for the development of the tool.

2. Related Work

In this section, we will discuss existing research related to model feedback and automatic grading, as well as their associated challenges.

In Literature [1], a tool is introduced that compares students' models with a model answer, automatically detects discrepancies, and then provides feedback to the students. However, one of the challenges highlighted is its inability to accurately account for synonyms.

Literature [2] allows for consistent automatic grading by comparing models using semantic, syntactic, and structural matching. Yet, a noted challenge is its failure to fully adapt to individual grading styles of instructors.

3. Requirements

We aim to create a tool that assists instructors in grading and providing feedback. Specifically, using natural language processing, we will examine similarities in class and attribute names, among other factors, to facilitate automatic grading. Based on the grading results, the tool will generate feedback explaining the reasons for any deductions.

The proposed features are as follows:

- Review function based on comparison with a model answer:
  - Point out vocabulary differences in class, attribute, and operation names. If different but semantically identical, consider it correct.
  - Highlight structural differences. If they differ but are semantically identical, consider it correct.
- Change Proposal: Rather than just pointing out vocabulary and structural differences, offer potential solutions and improvement suggestions.
- Common Error Pattern Warning: This feature detects and alerts users if common mistakes or misconceptions are identified during class diagram creation.
- Reference Material Links: Provide links to reference materials or literature to help users understand the theories or best practices behind the comments and suggestions.
- Flexible Settings: An option that allows instructors to customize the severity and criteria of comments.

4. Implementation

The current implementation of our proposed features is being developed as a plugin for the UML modeling tool Astah*. We have only realized the "Review function based on comparison with a model answer." This Astah* plugin transforms the model drawn by the learner into a text format. We have adopted the Mermaid format for this purpose. The model, expressed in the Mermaid format, is queried against ChatGPT alongside the model answer, and the feedback is then displayed.

We will explain the status of the implementation. Figure 1 shows a model drawn by a hypothetical student for demonstration purposes, while Figure 2 presents the model answer.

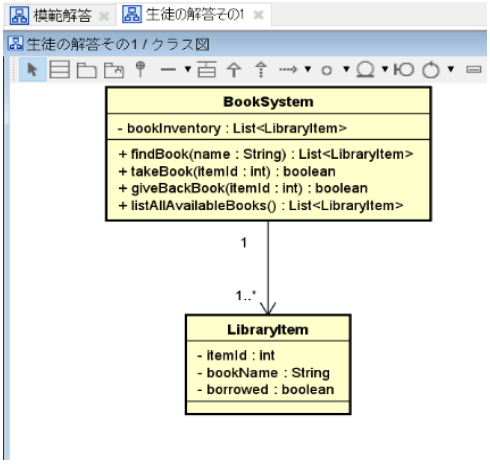

Figure 1. A Class Diagram for the Book System Assume to be Drawn by the Student.

†1 Shibaura Institute of Technology

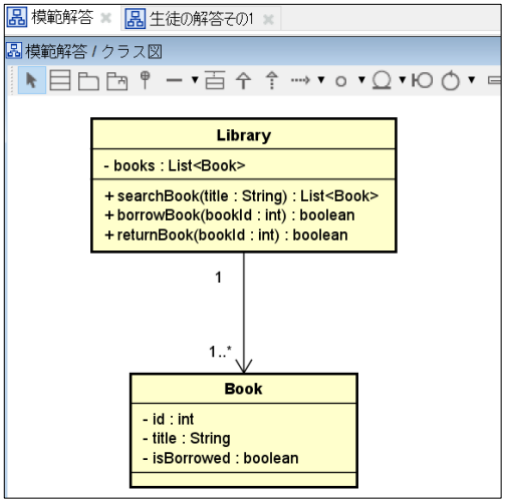

Figure 2. An Ideal Answer of the Book System

Firstly, this model drawn by students is converted into the Mermaid format, as shown in List 1. This format is then inputted into ChatGPT, along with the model answer, which underwent the same conversion. ChatGPT will then point out similarities and differences. As discussed in the requirements, even if terms differ, they will be noted as acceptable if they have similar meanings. An example of such feedback is displayed in Table 1.

Table 1 lists the differences between the *Answer* that can be considered the ideal or model solution and the *Student*'s submission in a UML design task using ChatGPT. The *Type* column specifies the UML element type, the *Answer* column indicates the ideal naming or representation, the *Student* column shows the student's provided naming or representation, and the *Reason* column highlights the reason for the discrepancy between the two.

For the *Class* type, while the ideal answer is *Library*, the student used *BookSystem*, indicating an omission. However, it should be deemed as similarity. The ideal answer for the Method is *searchBook*, whereas the student's answer is *findBook*. The reason for this difference is due to their similarity. Regarding the *Attribute* type, the ideal naming for one attribute is *Title*, but the student wrote *Name*, with the difference being their similarity. For another attribute, while the ideal answer is *isBorrowed*, the student used *borrowed*, also due to their similarity. Lastly, for the *Relation* type, the correct relationship is 'Library(1) > (1..)Book', but the student represented it as 'BookSystem(1) > (1..)LibraryItem', showcasing an *inaccuracy* in their understanding or representation. While the student used *BookSystem* and *LibraryItem* in contrast to the model answer's Library and Book, it can be argued that these terms, although different, are not incorrect but alternative representations or terminologies that capture similar concepts.

5. Conclusion

This paper has put forth a pioneering approach to enhancing UML modeling education by leveraging the capabilities of Gen AI. Our proposed UML model review tool not only identifies discrepancies between a student's model and a model answer but also appreciates the semantic nuances behind varying terminologies. Recognizing that differences in naming conventions can still represent similar underlying concepts, the tool provides precise and context-aware feedback. Beyond simple error spotting, the system offers potential solutions and improvement suggestions, ensuring students receive constructive and actionable insights. As UML modeling continues to be a critical skill in system design, tools like the one proposed here will play an indispensable role in training the next generation of system designers. This research serves as a foundational step in blending the strengths of AI with the intricacies of UML design, promising a richer and more effective educational experience.

In moving forward, there are key challenges to address. Firstly, allowing for more customization in feedback will cater to a broader range of student needs. Secondly, as diagrams become more complex, our tool's capability to review intricate UML designs will need refinement. Lastly, enhancing the tool's ability for deeper semantic understanding will be crucial for more nuanced feedback.

Reference

[1] Prabhsimran Singh, Younes Boubekeur, Gunter Mussbacher, "Detecting Mistakes in a Domain Model," MODELS '22: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 257–266, 2022.

[2] Weiyi Bian, Omar Alam, Jorg Kienzle, "Automated Grading of Class Diagrams," 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 700–709, 2019.

List 1. A Mermaid Formatted Model Converted from Figure 1

```
classDiagram StudentAnswer1 {
  class BookSystem {
    -bookInventory : List<LibraryItem>
    +findBook(name : String) : List<LibraryItem>
    +takeBook(itemId : int) : boolean
    +giveBackBook(itemId : int) : boolean
    +listAllAvailableBooks() : List<LibraryItem>
  }
  class LibraryItem {
    -itemId : int
    -bookName : String
    -borrowed : boolean
  }
    BookSystem(1) --> (1..*)LibraryItem
}
```

Table 1. A Feedback Example of the System (Formatted)

Type	Answer	Student	Reason
Class	Library	BookSystem	Omission
Method	searchBook	findBook	Similarity
Attribute	Title	Name	Similarity
Attribute	isBorrowed	borrowed	Similarity
Relation	Library(1) > (1..*)Book	BookSystem(1) > (1..*)LibraryItem	Inaccuracy