Hints based Approach for UML Class Diagrams

Sehrish Abrejo[1†], Amber Baig[2††], Adnan Asghar Ali[3†††], Mutee U Rahman[4††††], and Aqsa Khoso[5†††††]

Sehr.abrejo@gmail.com
amber.baig@isra.edu.pk
adnan_asghar_ali@hotmail.com
muteeurahman@gmail.com
sanamkhoso78@gmail.com

[1†,2††,3†††,4††††,5†††††] *Isra University, Hyderabad, Pakistan*

Abstract

A common language for modelling software requirements and design in recent years is Unified Modeling Language (UML). Essential principles and rules are provided by UML to help visualize and comprehend complex software systems. It has therefore been incorporated into the curriculum for software engineering courses at several institutions all around the world. However, it is commonly recognized that UML is challenging for beginners to understand, mostly owing to its complexity and ill-defined nature. It is unavoidable that we need to comprehend their preferences and issues considerably better than we do presently in order to approach the problem of teaching UML to beginner students in an acceptable manner. This paper offers a hint based approach that can be implemented along with an ordinary lab task. Some keywords are heighted to indicate class diagram component and make students to understand the textual descriptions. The experimental results indicate significant improvement in students learning skills. Furthermore, majority of students also positively responded to the survey conducted in the end experimental study.

Keywords:
Software Modeling, Unified Modeling Language, Class diagrams, ill-defined domain.

1. Introduction

The process of creating software is intricate and frequently surprising. Multiple teams must collaborate and plan together to create complex software, thus they must be able to communicate with each other clearly and succinctly. For this purpose, Unified Modeling Language (UML) was released by Object Management Group (OMG) back in late twentieth century. One of UML's objectives was to give the development community a table and a standard design language for making and maintaining business applications [1]. UML is a graphical language rather than a programming language and is linked with object oriented concepts.

The use of UML provides fundamental guidelines and conventions for visualizing and comprehending large software systems. Students will benefit from being taught how to design complicated requirements that software developers can understand if they follow the rules and

provide the artefacts [2]. This is the rationale for why UML is now included in many software engineering curriculum at universities all around the world [3][4]. However, UML poses challenges in the classroom and is rather controversial. UML is categorized as an ill-defined domain [5-7] because there are many model solutions to one scenario/requirement as compared to the well-defined domain where one scenario can be solved in only one way, i.e. there is only one solution to a problem. Furthermore, students are not experts, and they need to practice lots of examples to become successful analysts. Lastly, students also face problems while modeling a solution for a scenario which could be incomplete and ambiguous [8].

Numerous research have been undertaken to look at the difficulties that undergraduate students have when comprehending and creating OO models, [9-11], which are explained in subsequent sections. This research attempts to specifically look into the issues students have when creating class diagrams, and providing them hints along with textual descriptions of the system. The following section describes difficulties students encounter while modeling class diagrams followed by object-oriented modeling. Evaluation method is discussed before the results. Conclusions are presented in the end.

2. Literature Review

While the conventional learning method of UML modeling in a classroom setting can suffice as an introduction to the concepts of OO analysis and design, students cannot acquire expertise in the domain by simply attending lectures. Human tutors try to provide individual help to each student by providing them with different tutorials, still human tutors must meet the demands of the entire class, and students may only receive modest personal assistance. Many resources, textbooks, and UML tutorials are available for students. Despite these resources, students fail to understand the development of high-quality OO modeling [9].

Researchers have conducted several studies to investigate the problems undergraduate students face while

understanding and designing OO models. [10] in their study, they found errors in students' class diagrams which include: missing operations in class diagrams, the misconception of relationship multiplicities, and incorrect use of inheritance.

[11] concentrated on the assignment of attributes to classes in class diagrams. In their investigation, they found that adding a class to represent an entity in the class diagram without connecting it to the other existing classes using instance variables was the most frequent design mistake.

The authors in [12] study investigated that 31% of diagrams were incomplete with missing classes, attributes and missing or inappropriately defined associations. [13] investigates different categories of difficulties perceived by the learners and found that many students face problems while constructing UML diagrams due to insufficient course material, crowded classrooms and lack of good textbooks and user unfriendly CASE (Computer-Aided Software Engineering) tools.

[14] looked at the difficulties associated with creating and modelling UML diagrams from the standpoint of diagrammatic representation and reasoning. Their findings demonstrate that students' abilities to quickly recognize graphical symbols and see and link pertinent visual components are hindered by UML notational features with low perceptual discriminability. Students (novices) cannot distinguish UML notational components that are excessively similar, in contrast to specialists who can with practice make much finer differences [15]. Such individuals experience learning difficulties and their perceptual processes slow down because they must work harder and memories elements.

In the literature, the focus is on identifying the students' modeling problems by providing them textual description of system and then evaluating their models. Our strategy is based on hints that come with textual descriptions, for the purpose of creating UML models, so that the students can comprehend the textual content.

3. Object-Oriented Modeling

An Object-Oriented approach is widely used in software development [16-19], and learning how to create high-quality OO software is a central subject in Computer Science and Software Engineering curricula. When OO first made its debut, it was used (only) as a programming language in the mainstream of software engineering. Following that, its influence grew to include Object-Oriented Design (OOD) as a model for software design, and it grew even further to include Object-Oriented Analysis (OOA). The same OO principles for system structure are applied when conducting requirements evaluation in OO analysis to define the concepts, attitudes, and associations in a problem domain.

OO systems are composed of Classes (including their structures and behaviors) and relationships among them. Relationships can have different names, multiplicities, and types (association, aggregation, composition, inheritance or dependency). Since these constructs exist largely independently of any programming language in OOA and OOD, several notational systems for representing OO models without the need for source code have been created. Today, UML is the most commonly used notation. UML is usually used in software engineering courses to teach OO analysis and design.

There are several types of diagrams in the UML, but class diagrams define the static structure of OO systems: the classes and relationships, hence are the most important in the learning of OO modeling. Class diagrams are conceptually similar to ER (entity-relationship) diagrams which are also used for data modeling, but class diagrams correspondingly support OO features including inheritance and behaviors [20]. Given that defective specifications are related to the failure of a large percentage of established systems, it is crucial to ensure the consistency and quality of conceptual models developed early in the system development process. For several system analysts, developing high-quality computational models is a difficult task [21]. UML-ITS focuses on teaching students how to create a UML class diagram to describe OO concepts found in informal software requirement descriptions in textual form. This form of exercise has been used consistently in introductory software engineering courses for many years with the assistance of human tutors. By posing additional problems and offering automated tutoring, the system has been designed to complement the current teaching program.

Let's use a basic illustration to demonstrate the method of creating a class diagram. The following is a summary of the targeted system given to the students:

> *Design a class diagram for a university. A University is known by its Name, Address, and Phone Number and has one or many Departments. Each department has a department id and name and offers different courses. Department has one or many persons, i.e. Students and Teachers. A Person can have a Name, Phone Number, Email Address and Address (including House number, Street, City, Postal Code and Country name). If the Person is a Teacher then it should contain Salary, and if the person is Student, then it should contain Student ID and Average Marks. One teacher can teach one or more courses and each course can have one or more students. Each course is known by its Course ID, Course Name and Credits Hours. A university can access information about departments. The students can enroll or drop several courses.*

From the above descriptions, students can identify different classes, such as *universities, Departments,*

Persons, Teachers, Students, Courses and Addresses. The students can also identify attributes and methods related to each class. For example, class University has Name, Address and Phone-Number as attributes and ShowDeptData() as the method. After all classes and their relative attributes and methods have been identified, students then identify the relationship types between all classes. For example, University has one or more departments, which is mentioned in the description. The students can also define the name and multiplicity of each relationship. The class diagram of the university system is illustrated in Figure 1.
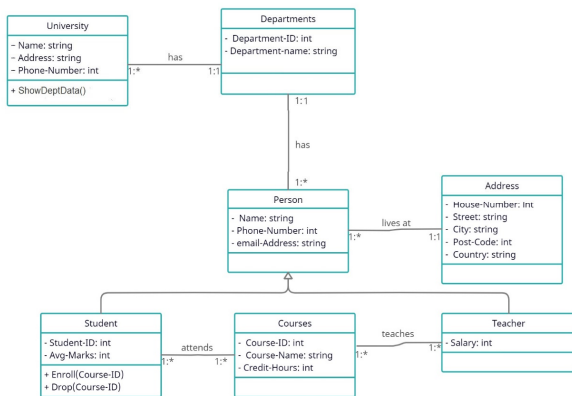
Figure – 1: University UML Class Diagram

It can be inferred from the above example description that students must be familiar with various concepts before developing a UML class diagram for any system. All classes, attributes, methods, and relationships are explicitly mentioned in the above example, but in real scenarios, the description text is mostly longer, incomplete and ambiguous. Furthermore, UML modeling is not a well-defined domain, UML modeling is an ill-defined domain, which means that there are many possible and correct solutions to a single problem. Students usually face many issues while learning how to construct a good quality OO model.

4. Methodology

This study's primary goal is to teach students to correctly comprehend and develop UML models. In order to investigate students' learning effects, following study design was created

4.1. Participants

For the experiment, total of 80 students who were enrolled in software engineering course in software engineering and computer science discipline from a local university participated. The students had little to no understanding of UML modeling because course is offered to the beginners. In order to obtain accurate findings, we picked these research options with the maximum level of contrast.

4.2. Experiment design

For the experimental purpose, UML class diagrams were selected as domain. It was four weeks activity. The 3 hours experiment started with the lecture of UML class diagrams modeling, followed by lab tasks to be performed by students in every week. In the lab tasks, students were asked to carefully read the textual descriptions of some software system and were asked to draw model diagram, except for 3rd experimental lab task which was based on reverse engineering, in which, students examined a model diagram and wrote textual descriptions. The students were seated in the same lab and were not allowed to interact with each other without instructor's permission. Students were asked to respond to a survey on their awareness of UML class diagrams at the end of the lab session 4. Every component of UML class diagrams was covered by the survey's 10 total items. The survey was based on 5 Likert scale options from strongly disagree to strongly agree. The survey can be seen in Figure 2.

Figure. 2 Class diagram Survey.

4.3. Types of lab tasks

For experimental purpose, four types of lab tasks for each week were designed to investigate the learning skills of students.

Week 1 lab task: consists of 5 textual descriptions of different systems covering different aspects of class diagrams. The text was plain and had no hint regarding class
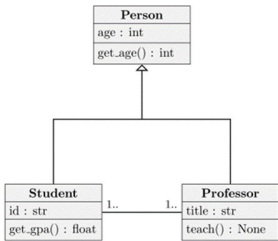
diagram components, which is usually the case followed in most institutes.

Week 2 lab task: same task consists of 5 textual descriptions with hints (underline/italic/bold words) to identify different components (class, attribute, method, relationship) of class diagrams.

Week 3 lab task: different task with 5 model diagrams. The students were asked to fill in the blanks with the help of hints to write the textual description of the system.

Week 4 lab task: different tab tasks with 5 textual descriptions of different systems covering different aspects of class diagrams. The text was plain and had no hint regarding class diagram components. Table 1 shows one task from each lab task.

Table 1: Types of Lab Tasks

Week	Type of Task
Week 1	An Artist can compose one or more albums. Each Album with Name and Downloading Link can contain many songs. Each song has name and length with it. A song can only belong to one album.
Week 2	An Artist can compose one or more albums. Each Album with <u>Name</u> and <u>Downloading Link</u> can *contain* many songs. Each song has <u>name</u> and <u>length</u> with it. A song can only *belong to* one album. Bold: Class Underline: Attributes/methods Italic: Relationships and multiplicities
Week 3	Fill in the blanks and Write a textual description of following diagram. Classes 1: (Hint: See Bold names in rectangles) _____ Attributes 1: (Hint: See second part of each rectangle) _____ . . . . . *(Similar blanks for methods and relationships)* Person age : int get_age() : int Student — Professor id : str / title : str get_gpa() : float / teach() : None

Week 4	E-document can be a simple Book or it can be an Email. They both have the author and date but book has title whereas email has subject and the sender name.

5. Results & Evaluations

Main aim of this study was to investigate the effects of manual hints provided in ordinary lab tasks, on students' learning. The investigation started with comparison of student's solutions of lab task 1 and lab task 4 to see the difference in their learning skills. Furthermore, the survey results also helped in evaluating the students' satisfaction with hints based approach for UML modeling.

5.1. Difference in learning skills.

The most important evidence of hints based approach success is an improvement in students' ability to learn. Figure 3 shows the average of correct and incorrect class diagram components based on the solutions that students modeled in the end of lab tasks1. It was found that 72% of students faced problem in locating class components from the description, 65% of students failed to assign correct attributes to their classes and 82% of students drew incorrect relationships among classes. On the other hand, 60% of students correctly identified class attributes from the textual description.

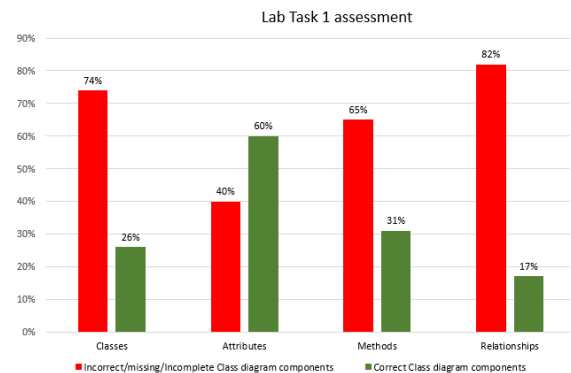Figure. 3 Assessment results of students' models in lab task 1.

After the experimental treatment of lab task 2 & 3, the learning skills of students' improved as shown in figure 4. It can be observed that students were able to find out classes, attributes, methods and relationships from textual descriptions in lab task 4. Furthermore, the error rate of each student also dropped which can be seen figure 5.
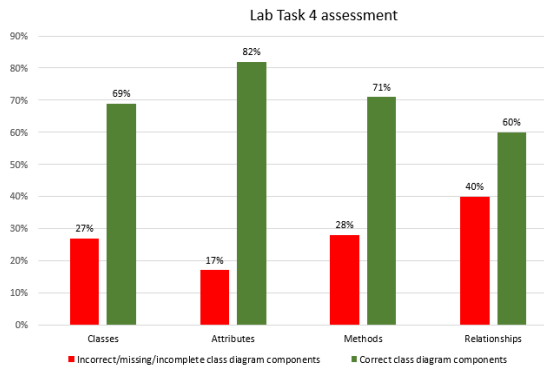
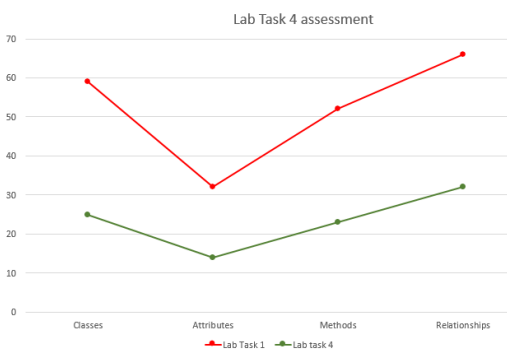Figure. 4 Assessment results of students' models in lab task 4.

Figure. 5: Difference among students' errors rate in lab task 1 & 4.

Furthermore, an independent sample-t test was performed to observer the difference in mistakes made by students in both tasks. The results shown in table 2 indicates the significant difference ($t = 3.500$, $p = 0.013$) in both conditions.

Table 2: Difference between error rates in Labtask1 and Labtask4.

Test Data	Mean	S.D.	Statistical Test	t-value	Sig(2-tailed) p value
LabTask1	52.25	14.6	Independent Sample t-test	3.500	0.013
LabTask4	23.5	7.4			

5.2. Class diagram survey

The satisfaction of the students with the hint-based strategy was determined by a survey that was administered at the end of the experimental activity. Survey responses are shown in figure 6. It can be observed that the majority of students were able to comprehend class diagrams from written descriptions. With the exception of items 6 and 10, up to 80% of students responded positively in replies to most of the survey items. Up to 40% of students faced problems in identifying relationships among classes, as shown in the survey items 6 & 10.
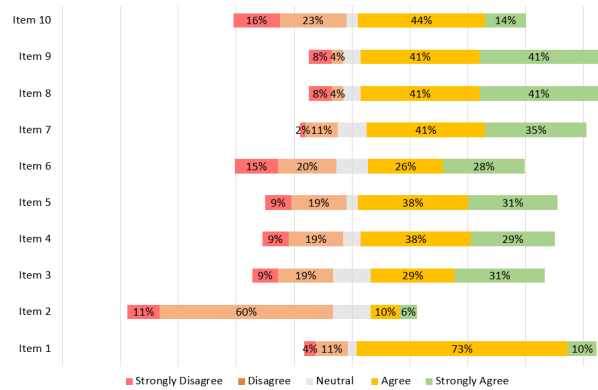

Figure. 6: UML Class Diagram Survey Responses.

6. Conclusion and Future work

UML is one of the general purpose modeling language that aims to provide a uniform method of visualizing system design. UML is valuable content to learn in contemporary courses on software engineering. Its intricacy makes it difficult for beginners to understand. This research attempts to investigate the effects on students learning skills if ordinary lab tasks are equipped with supporting material, i.e. hints. Students' learning abilities improved after four weeks of investigative work. When the students received suggestions for their lab assignments, their mistake rate drastically decreased. The survey that was administered as part of the final activity received a favorable response from the students as well. Thus, it may be inferred that students will learn more effectively if routine lab exercises include hints.

In addition to the assistance given to students in completing their tasks, it is important to carefully analyze the issues they encounter. According to the survey results, students continue to have difficulty modelling relationships between classes in class diagrams. In the future, this research will attempt to best help students in order to simulate linkages across classes. In addition, our next research will also analyze the difficulties students have modelling various UML diagrams, such as use cases, state machines, etc. It is undeniable that a substantial amount of research is being done to aid students in understanding UML modelling diagrams, but this is still a challenging task and an unexplored domain.

References

[1] Bell, D. (2003). UML basics: An introduction to the Unified Modeling Language.

[2] Reuter, R., Stark, T., Sedelmaier, Y., Landes, D., Mottok, J., & Wolff, C. (2020, April). Insights in students' problems during UML modeling. In 2020 IEEE Global Engineering Education Conference (EDUCON) (pp. 592-600). IEEE.

[3] "Computing curricula 2001," ACM, IEEE, Tech. Rep. 3es, 2001.

[4] "Computing Curriculum - Software Engineering," ACM, IEEE, Tech. Rep. 0003263, 2004.

[5] Baker, M. M., New, A., Aguilar-Simon, M., Al-Halah, Z., Arnold, S. M., Ben-Iwhiwhu, E., ... & Vallabha, G. K. (2023). A domain-agnostic approach for characterization of lifelong learning systems. Neural Networks, 160, 274-296.

[6] Gross, S., Mokbel, B., Hammer, B., & Pinkwart, N. (2015). Learning Feedback in Intelligent Tutoring Systems: Report of the FIT Project, Conducted from December 2011 to March 2015. KI-Künstliche Intelligenz, 29, 413-418.

[7] Lukyanenko[1], R., Parsons, J., & Storey, V. C. (2023, May). Check for updates Principles of Universal Conceptual Modeling Roman Lukyanenko[1] (), Jeffrey Parsons[2], Veda C. Storey[3], Binny M. Samuel, and Oscar Pastor5 d. In Enterprise, Business-Process and Information Systems Modeling: 24th International Conference, BPMDS 2023, and 28th International Conference, EMMSAD 2023, Zaragoza, Spain, June 12–13, 2023, Proceedings (Vol. 479, p. 169). Springer Nature.

[8] Baghaei, N., Mitrovic, A. and Irwin, W. A., (2005), "Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML", In Looi, C., Jonassen, D. and Ikeda M. (Eds.), ICCE, 2005, pp.11-18

[9] Shen, Z., Tan, S. And Siau, K., (2018), "Challenges in Learning Unified Modeling Language: From the Perspective of Diagrammatic Representation and Reasoning", Communications of the Association for Information Systems, pp. 545–565

[10] Bolloju, N. and Leung, F., (2006), "Assisting Novice Analysts in Developing Quality Conceptual Models with UML", Communications of the ACM, 49, pp. 108–112.

[11] Thomasson, B., Ratcliffe, M. and Thomas, L., (2006), "Identifying Novice Difficulties in Object Oriented Design", ACM SIGCSE Bulletin, 38, pp. 28–32.

[12] Ven Yu Sien, V. Y., (2011), "An Investigation of Difficulties Experienced by Students Developing Unified Modelling Language (UML) Class and Sequence Diagrams", Computer Science Education, 21(4), pp. 317–342.

[13] Siau, K., and Loo, P. P., (2006), "Identifying Difficulties in Learning UML", Information Systems Management, 23(3), pp. 43-51.

[14] Shen, Z., Tan, S. and Siau, K., (2018), "Challenges in Learning Unified Modeling Language: From the Perspective of Diagrammatic Representation and Reasoning", Communications of the Association for Information Systems, pp. 545–565.

[15] Britton, C. and Jones, S., (1999), "The untrained eye: How Languages for Software Specification Support Understanding by Untrained Users", Human Computer Interaction, 14(1), pp. 191-244.

[16] Sommerville, I., (2004). Software Engineering. Pearson/Addison-Wesley, 7th ed.

[17] Kalinga, E. A. (2021). "Learning Software Development through Modeling Using Object Oriented Approach with Unified Modeling Language: A Case of an Online Interview System". Journal of Learning for Development, 8(1), 74-92

[18] Al-Msie'deen, R. F., and H Blasi, A. (2021). "Software Evolution Understanding: Automatic Extraction of Software Identifiers Map for Object-Oriented Software Systems". Journal of Communications Software and Systems, 17(1), 20-28.

[19] Kaur, S., Awasthi, L. K., and Sangal, A. L. (2021). "A review on software refactoring opportunity identification and sequencing in object-oriented software". Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering), 14(3), 252-267.

[20] Booch, G., Rumbaugh, J. and Jacobson, I., (1999), "The Unified Modelling Language User Guide", Reading: Addison-Wesley.

[21] Bolloju, N. and Leung, F., (2006), "Assisting Novice Analysts in Developing Quality Conceptual Models with UML", Communications of the ACM, 49, pp. 108–112.
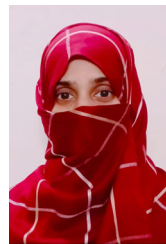
Sehrish Abrejo received her Bachelor's of Computer Science degree from Isra University, Hyderabad. She continued her further studies in the same discipline and received MSCS, M.Phil and PhD degrees from same institute. She is currently working as Assistant Professor in the department of Computer Science at Isra University, Hyderabad, Pakistan. Her research areas are Mobile ad hoc Networks, Artificial Intelligence and Natural Language Processing.

Mutee U Rahman received the B.Sc. and M.Sc. degrees in Computer Science, from University of Sindh in 1997 and 1999, respectively. He received the PhD. In Computer Science degree from Isra Univ. in 2017. He is working as a Professor in Computer Science at Isra University, Hyderabad Pakistan. His research interests include: Natural Language Processing, Computational Linguistics and Artificial Intelligence.

Adnan Asghar is currently pursuing his Ph.D. degree in Information Technology at University of Sindh, Jamshoro, Pakistan. Presently working as Senior Lecturer in the Department of Computer Science at Isra University, Hyderabad, Pakistan. His research interests include: Internet of Things (IoT), Network Science, Technology and Innovation Management.

Amber Baig received the BS and MS degrees in Computer Science from IMCS, University of Sindh, Jamshoro and the M.Phil and PhD degrees in Computer Science from DCS, Isra University, Hyderabad. She is currently working as Associate Professor in the Department of Computer Science, Isra University, Hyderabad, Pakistan. Her research interest includes Artificial Intelligence, Natural Language Processing and Human Computer Interaction.

Aqsa Khoso is an accomplished Teaching Assistant and a dedicated student currently pursuing a Master's degree in the Department of Computer Science at Isra University, Hyderabad. Throughout her academic journey, she has developed a profound interest in the captivating field of Natural Language Processing (NLP).