computers

Article

Learning from Peer Mistakes: Collaborative UML-Based ITS with Peer Feedback Evaluation

Sehrish Abrejo *, Hameedullah Kazi, Mutee U. Rahman, Ahsanullah Baloch and Amber Baig

Department of Computer Science, Faculty of Engineering, Science & Technology, Isra University, Hyderabad 71500, Pakistan; hkazi@isra.edu.pk (H.K.); mutee.rahman@isra.edu.pk (M.U.R.); ahsanullah.baloch@isra.edu.pk (A.B.); amber.baig@isra.edu.pk (A.B.)
* Correspondence: sehrish-abrejo@hotmail.com

Abstract: Collaborative Intelligent Tutoring Systems (ITSs) use peer tutor assessment to give feedback to students in solving problems. Through this feedback, the students reflect on their thinking and try to improve it when they get similar questions. The accuracy of the feedback given by the peers is important because this helps students to improve their learning skills. If the student acting as a peer tutor is unclear about the topic, then they will probably provide incorrect feedback. There have been very few attempts in the literature that provide limited support to improve the accuracy and relevancy of peer feedback. This paper presents a collaborative ITS to teach Unified Modeling Language (UML), which is designed in such a way that it can detect erroneous feedback before it is delivered to the student. The evaluations conducted in this study indicate that receiving and sending incorrect feedback have negative impact on students' learning skills. Furthermore, the results also show that the experimental group with peer feedback evaluation has significant learning gains compared to the control group.

Keywords: Intelligent Tutoring Systems; UML class diagrams; peer feedback evaluation; domain learning

check for updates

Citation: Abrejo, S.; Kazi, H.; Rahman, M.U.; Baloch, A.; Baig, A. Learning from Peer Mistakes: Collaborative UML-Based ITS with Peer Feedback Evaluation. *Computers* **2022**, *11*, 30. https://doi.org/10.3390/computers11030030

Academic Editors: Antonio Sarasa Cabezuelo and Covadonga Rodrigo San Juan

Received: 31 December 2021
Accepted: 16 February 2022
Published: 22 February 2022

1. Introduction

ITSs are computer-based learning systems that use Artificial Intelligence (AI) techniques to simulate human tutors to help students to improve their learning skills. Today, ITSs are in widespread use at various levels in different advanced countries and are enhancing the student learning experience [1–3]. ITSs have been successfully developed for a variety of domains including mathematics, physics, programming, databases, design tasks and learning new languages. Examples include Andes Physics Tutor (problem solving in introductory college physics) [4], Algebra Cognitive Tutor (problem solving in a high school algebra course) [5], AutoTutor (problem solving in college physics and other domains) [6,7], Sherlock (troubleshooting a large piece of simulated electrical equipment) [8], SQL-Tutor [9], COLLECT-UML (Object-Oriented software design) [10–12] and KERMIT (database design) [13–15].

With the advent of the Internet, students may now not only attend school to listen to live lectures, but they can also use Internet platforms to develop skills. To put it another way, online learning can help students study more efficiently [16]. ITSs use cutting-edge computer technologies such as the Internet, hypermedia and virtual reality to provide tutoring for individual students, and groups of students, in a collaborative learning environment [17]. Collaborative ITSs use peer feedback where the learner receives feedback from other students. Feedback can be defined as "all post-response information that is provided to a learner to inform on his or her actual state of learning or performance" [18,19]. The source of feedback can be external (peer or teacher) or internal (the learner). Peer feedback is a practice where students give feedback to each other, hence improving their learning in a particular domain. Learning of students, while providing feedback, is improved when they are involved in talking, listening, writing, reading, and reflecting on contents, ideas, and problems in the domain. Researchers have widely reported the benefits of peer tutoring and feedback in classroom settings. Refs. [20–24] present hundreds of experiences in the field. Recent literature reviews in [25–27] document the academic, social, and psychological benefits of this methodology.

In collaborative learning, students of the same learning status provide feedback to each other. The feedback can be in the form of formative assessment, which is equivalent to that of teacher's feedback [22]. In formative assessment feedback, the main difference between teacher's and peer student's feedback is that the peer is not an expert of the domain; as a result, the feedback of peers varies. Not all peer feedback results in learning gains, and the students should receive relevant feedback as guidance to revise their solutions rather than confuse them [28]. There is always a chance that the peer may not be clear about the topic and may provide wrong information. On the other hand, the student receiving inappropriate feedback can infer incorrectly, which leads to having a negative impact on students' learning. In contrast, any misinterpretation of peer feedback will cause learners to lose focus and take longer to comprehend and solve the problem. Many collaborative systems have been proposed in the literature in which one student's solution is evaluated by another student. Examples of such systems include Collab-ChiQat [29], CirCLE [30] and ITSCL [31]. In these systems, one student provides feedback on another student's solution, by rating or commenting. These ratings or comments can be misleading if they are contrary to the solution; for example, if the student has correctly solved the given problem but receives a negative rating from other students. Negative feedback can confuse students and they will not be able to solve the problem. Previous studies that allow collaboration through feedback from one student to another have revealed little or no evidence on peer feedback relevancy to correct/wrong solutions. Hence, considering the importance of peer feedback, it can be hypothesized that students receiving incorrect or irrelevant feedback from peers will have a negative impact on their domain learning. Therefore, the research question that we investigated in this study is whether there is a negative impact on students' learning if they receive incorrect feedback. To investigate and identify the effects of peer feedback on students' learning, this paper presents a Unified Modeling Language-Intelligent Tutoring System (UML-ITS) with peer feedback evaluation that focuses on improving the accuracy and relevancy of feedback in accordance with the solution. The presented ITS model not only evaluates peer feedback before delivering it, but also guides students in providing information that is correct and can be applied to the solution. The following section describes some of the ITSs that support collaboration. Next, the UML-ITS model along with its architecture and interface is discussed, before the presentation of the results. Finally, conclusions are presented.

2. Literature Review

The ITS has become increasingly common in assisting students [32]. In the literature, many tutoring systems have been proposed to support collaborative learning within ITSs to enhance students' learning. Collect-UML [12] is an ITS based on constraints that teach object-oriented (OO) analysis and design using UML. Students can use this tool to solve problems both alone and in collaboration. First, students use the system's feedback to develop UML class diagrams on their own. Then they join a group to come up with a group solution. The system compares the group solution against the individual solutions of all group members in collaboration mode. System feedback is provided on the group solution, and on the collaboration that takes place between students.

Collab-ChiQat [29] supports paired programming with a group of two students. Collab-ChiQat is designed to help students to learn linked list data structures in collaboration. One student can take a turn as a driver (the role assigned to one who writes the lines of code) at a time, and another student has to wait for his turn to write the code. The ITS provides domain hints to help the driver student to solve the given problem while another student can provide helpful feedback using a peer feedback bonus from the collaboration panel interface.

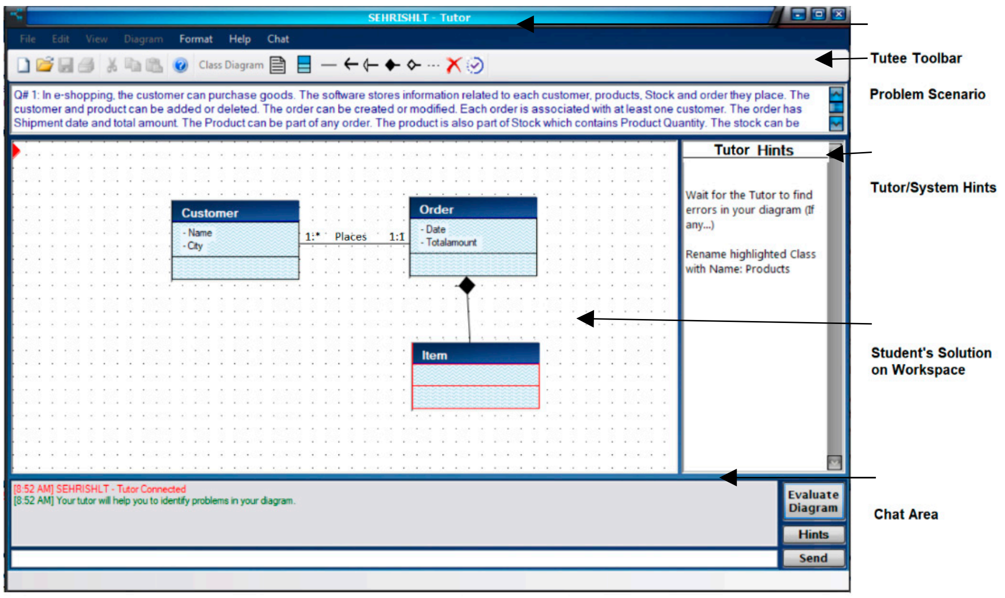
CirCLE [30] is the abbreviation for Circuitously Collaborative Learning Environment, which is designed for mathematical word problems to enhance the metacognitive awareness of the learning process. The main aim of this research was to make students rethink their solved problems after reviewing other students' solutions. Initially, all students are given a single problem which is then submitted to the system. After the first submission, the students are assigned the role of Peer Inspector to review and comment on other students' solutions. The inspector might consider rethinking their solution after receiving appropriate feedback on it and reviewing other solutions (metacognitive awareness).

ITSCL stands for Intelligent Tutoring Supported Collaborative Learning, which was proposed by [31]. It is a combination of an Intelligent Tutoring System and collaborative learning. ITSCL provides learners with three levels of interaction with the system. In the first level of interaction, the individual learner interacts with the ITS tutor without any collaboration. In this level of interaction, a single student uses the ITS on a one-on-one basis where the ITS asks questions of the student and the student has to respond. The intelligent hints are provided by ITS if the student is not able to answer or needs help. The second and third levels of interactions support learner-to-learner (two peer students) and tutor-group collaborative learning, respectively. In the second level of interaction, two students collaborate via a chat interface and share ideas to answer the questions. Hints are provided by the ITS based on their answers given. The third level of interaction allows tutor-group learning, where multiple students provide a group answer to the problem given by the ITS. In this form of interaction, each student provides an individual answer to the question given by the ITS, and the ITS displays each student's response to all other students in the group. Each student can update his/her response twice after reading other students' answers. After updating, students finalize their answers. The ITS uses natural language processing (NLP) on the finalized answers and matches the similarity with the answers stored in the database. The answer that most closely matches the stored database answer is selected as the group answer and ITS hints are then provided by the system on the selected group answer.

These systems support multiple students to collaboratively solve a problem but they do not evaluate the feedback that students send to each other while solving problems. In Collect-UML, the system does not restrict students from adding wrong elements in the collaborative group diagram. The students receiving help from the group diagram may implement wrong elements in their individual solutions. Collab-ChiQat allows the non-driver student (who is not writing the code) to comment or provide helping hints to the driver student. The feedback provided by the non-driver student can be irrelevant as they select already defined feedback from the drop-down list. ITS provides hints only when the driver student types erroneous code. In CirCLE, the inspector can disagree on students' solutions and can provide negative feedback despite all correct steps being taken. There is no support provided by the system to inform the inspector about the student's correct solution. Similarly, In ITSCL, each student can comment on the other student's answer. They can agree or disagree with the peer answer, or they can suggest changes in the peer answer. These comments are not evaluated by the ITSCL if one student provides false negative comments (the answer is correct but students comment negatively) or false positive comments (the answer is wrong but students comment positively. Moreover, textual feedback can also be helpful [33–35], but domain-related accuracy and relevance of the feedback delivered in natural language needs considerable usage of NLP techniques; otherwise, the system's ability to provide meaningful feedback would be debated. Furthermore, systems such as Collab-ChiQat and CirCLE bound students to submit feedback within the interface's boundaries, making it impossible for them to expound upon or explain their ideas. This research attempts to overcome the limitations of previous systems and proposes a peer feedback evaluation model in ITSs that examines peer feedback before it is sent to another peer. Peer feedback evaluation will not only help students to focus on providing responses that are relevant to the solution, but also help peers to rethink their feedback and avoid repeating the same mistakes while responding on similar errors.

3. System Description

Due to its complex and ill-defined nature, the UML Class Diagram was selected as a domain for the ITS design. The proposed UML-ITS is implemented to teach OO analysis and design concepts, where students collaboratively construct a UML class diagram based on some given requirements. UML-ITS creates a session between two students and assigns them roles of Tutor (the one who evaluates the solution of the other student and provides suggestions/feedback) and Tutee (the one who develops the solution). For the rest of the paper, the role names of Tutor and Tutee are used. Each action performed by the student is recorded in the log file. The students develop a solution model by drawing diagrams on the workspace area and interact with each other using the chat tool. UML-ITS provides feedback to students while solving a problem through hints. Once both students agree on the modeled solution, they can submit the final solution to the ITS. The solution is evaluated against the sample/ideal solution stored in the knowledge base. If the submitted solution has errors, the hints are generated; otherwise, a new problem scenario is displayed. Figure 1 shows the interface of UML-ITS. The same screen is displayed to both students but with different toolbar options.



Q# 1: In e-shopping, the customer can purchase goods. The software stores information related to each customer, products, Stock and order they place. The customer and product can be added or deleted. The order can be created or modified. Each order is associated with at least one customer. The order has Shipment date and total amount. The Product can be part of any order. The product is also part of Stock which contains Product Quantity. The stock can be

Wait for the Tutor to find errors in your diagram (If any...)

Rename highlighted Class with Name: Products

[8:52 AM] SEHRISHLT - Tutor Connected
[8:52 AM] Your tutor will help you to identify problems in your diagram.

Figure 1. UML-ITS interface.

3.1. XML Solutions

XML stands for eXtensible Markup Language, which is widely being accepted as a form of information representation. XML stores information in the form of tags. In UML-ITS, all the ideal solutions of UML textual problems are stored in XML documents. All XML documents begin with an XML declaration followed by a root element, which is <ClassDiagram> in our case. Each class is defined with the <Class> tag, which includes <Name>, <Attribute>, <Method>, and <Relationship> tags as child elements. The <Relationship> tag has four attributes:

1. WithClass: Defines the name of the class that has a relationship with the current class.
2. Type: Defines the type of relationship, i.e., Association, Generalization, etc.
3. Link: Defines Start or End of the line connector. This helps in the identification of parent-child and whole-part relationships if generalization, composition, or aggregation types are used. The Link = "End" attribute value represents the "Whole" class relationship, whereas Link = "Start" represents the "Part" class relationship. For class inheritance, the parent class is represented with the End link type and child class with the Start link type.
4. Multiplicity: Defines participation constraints on association relationship types.

The student's solution is also converted to a temporary XML document that is compared against the XML solution stored in the UML-ITS's knowledge base. The temporary XML document is updated whenever the tutee student makes changes in the solution. If the temporary XML document is different from that of XML solution, the hints are generated by the UML-ITS based on the differences found. Figure 2 describes the comparison of both XML documents.

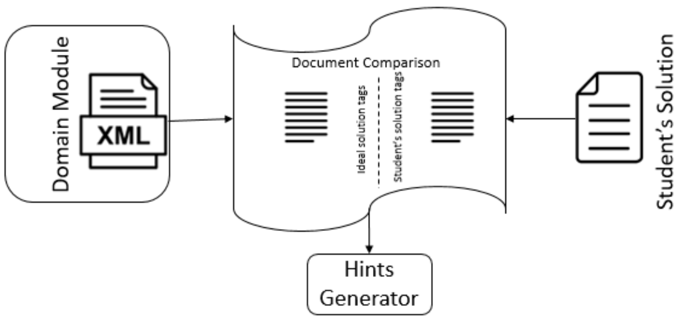


Figure 2. Ideal solution comparison with the student's solution.

3.2. Tutee Student Toolbar

The tutee student will start drawing the UML diagram by selecting the appropriate tool from the UML constructs (Figure 3).

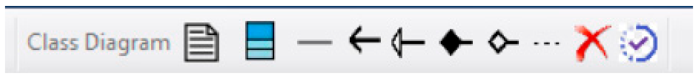


Figure 3. Tutee student toolbar.

To draw a diagram, the tutee student clicks on a specific drawing tool button, places the cursor on the desired workspace location, and presses the mouse button; for example to create a new class, the tutee clicks on the Class button (▤). An empty class component is displayed in the workspace area. The tutee then can define the name of the newly created class by double-clicking on the top portion of the class. The same method is used to change/modify the name of the class already present in the diagram. Once the class is created, its attributes and methods can be defined by double-clicking on the specific class component. The student can also change or delete class attributes/methods by right clicking and selecting required options. The relationships can be added by selecting appropriate relationship types from the toolbar.

3.3. Tutor Student Toolbar

The tutor student has the responsibility to identify tutee's misconceptions about UML class diagrams in their designed solution. Once the tutee student clicks the Evaluate Diagram button, tutor-student toolbar features are activated with the system-generated message to ask tutor to find errors in the diagram. To help tutor to find errors, some specific features are added to the tutor's toolbar, as shown in Figure 4.

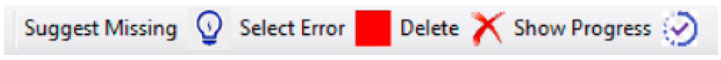


Figure 4. Tutor student toolbar.

The tutor can use different toolbar features to indicate the type of error in the tutee's solution. The Suggest Missing button (💡) can be used for missing components (classes, attributes, methods or relationships), the Select Error button (🟥) for incorrect components, and the Delete button (✗) for extra components. If the tutor finds some errors in the diagram, then the tutor can indicate them by activating the Select Error button. The tutor can only click on specific diagram components to indicate an error. The selected component's color is changed to red, which is also visible to the tutee. Furthermore, an automatic message is generated and sent to the tutee that contains information about the error and is displayed in UML hints area on the screen.

3.4. Peer Feedback Evaluator Design

The evaluation of tutor feedback takes place before delivering it to the tutee student. The main function of the feedback evaluator module is to evaluate tutor feedback against the ideal solution. The first step in evaluating the tutor's feedback is to identify the type of feedback that the tutor is providing to the tutee student. The sub-components of the Feedback Evaluator are shown in Figure 5.

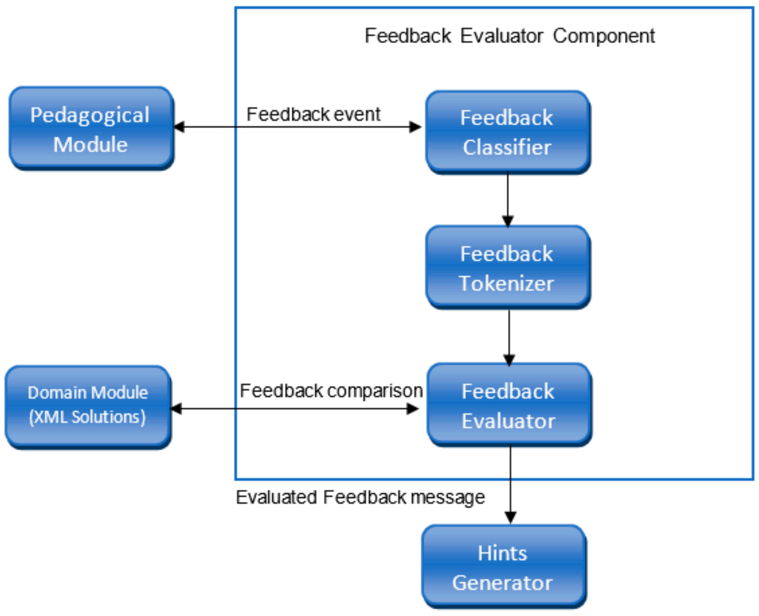


Figure 5. Sub-components of the feedback evaluator.

It can be seen in Figure 5 that the feedback classifier receives a feedback event, which is the hint that the tutor is providing to the tutee. The feedback classifier identifies whether it is related to classes, relationships, attributes, or methods. This identification is based on the tutor's actions that he/she has taken on the workspace. If the tutor clicks the class component, then it is classified as 'C', relationships as 'R', attributes as 'A', and methods as 'M'. For example, if the tutor finds an error in the solution related to classes, then the tutor clicks on that particular class. As soon as the tutor clicks on the class, the following message is generated:

C ClassName = "Items"

The message indicates that the tutor has clicked on the class component and its name is 'Items'. Once the component on which the tutor is providing feedback has been identified, then further classification takes place to see if the response is related to a wrong component, a missing component, or extra classes in the tutee's solution. This classification is based on the button activated from the tutor's toolbar. If Suggest Missing is activated, then a token related to the missing component is added to the feedback; if the Error button is selected, then a token for the wrong component is added; and if the Delete button is activated, then an extra component token is added to the feedback message. The feedback tokenizer assigns tokens based on their classification, as shown in the following example, which indicates that the 'Items' class is an incorrect class in the tutee's solution.

WrongC ClassName = "Items"

For attributes and methods, WrongA and WrongM tokens are used, respectively. Along with the class name in which the wrong attribute or method is defined, the attribute and method names are also appended in the feedback message. For example, the following feedback messages indicate that the tutor has marked the attribute with the name 'id' in the class 'Items' as a wrong attribute. Similarly, the method with name 'Show' in the class 'Items' has been marked as a wrong method.

WrongA ClassName = "Items" Attribute = "id"

WrongM ClassName = "Items" Method = "Show"

For errors in relationships, more information is added in the message to indicate the endpoints' directions along with the type of relationships. For example, if the tutor finds an error in the composition relationship type, then the following feedback message is generated:

WrongR ClassName = "Items" WithClass = "Orders" Type = "Composition" Link = "Start"

The above message indicates that the relationship of the type Composition between the Items and Orders classes is wrong. The endpoints of relationships are identified through the Link attribute. If the endpoints of the relationship are drawn incorrectly, then it is also considered to be an error in the relationship, which is tokenized similarly, as described above. Table 1 shows the list of tokens that are assigned to feedback messages.

Table 1. List of Tokens.

Token	Description	Token	Description
WrongC MissingC ExtraC	Wrong Class in Tutee's solution Missing Class in Tutee's solution Extra Class in Tutee's solution	ClassName	Name of Wrong/Missing/Extra class
WrongA MissingA ExtraA	The wrong attribute in Class Missing attribute in Class Extra attribute in Class	Attribute	Name of Wrong/Missing/Extra attribute
WrongM MissingM ExtraM	Wrong Method in Class Missing Method in Class Extra Method in Class	Method	Name of Wrong/Missing/Extra method
WrongR MissingR ExtraR	Wrong Relationship b/w classes Missing Relationship b/w classes Extra Relationship b/w classes	With Class TypeLink	Second class name Name of relationship End connectors of relationship

The feedback evaluator compares the tokenized feedback message with an ideal solution to see if the tutor is responding correctly to the tutee. This comparison is based on following conditions:

- A missing class diagram component is present in the ideal solution but not in the tutee's solution.
- A wrong class component is present in the tutee's solution but not in the ideal solution.
- An extra class diagram component is present in the tutee's solution but not in the ideal solution. This condition also checks if the total number of classes present in the tutee's solution is greater than the total number of class components in the ideal solution. If the number is greater, then the selected component is considered to be extra; otherwise, it is considered to be wrong.

Once the feedback evaluator compares the feedback with the ideal solution, extra information is added in the feedback message. If the tutor has correctly marked an incorrect class diagram component as the wrong class, then a plus (+) sign is added at the beginning of the message, which indicates that the feedback from the tutor is correct. Conversely, if the tutor has marked a correct class diagram component as the wrong class, then a minus ($-$) sign is added at the beginning of the feedback message to indicate incorrect feedback from tutor.

Table 2 shows some of the examples of correct and incorrect feedback message identification. The feedback message along with token and sign is delivered to the Hints generator to produce hints accordingly.

Table 2. Correct/incorrect feedback messages.

Correct Tutor Feedback
+WrongC ClassName = "Items"
+ExtraA ClassName = "Items" attribute = "id"
+MissingM ClassName = "Items" method = "Show"
+WrongR ClassName = "Items" WithClass = "Orders" type = "Composition" Link = "Start"
Incorrect Tutor Feedback
−MissingC ClassName = "Orders"
−ExtraA ClassName = "Items" attribute = "id"
−MissingA ClassName = "Items" attribute = "Price"
−ExtraR ClassName = "Items" WithClass = "Products" type = "Association" Link = "Start"

3.5. Tutor Feedback Evaluation Model

Tutor feedback in the UML-ITS is evaluated by the feedback evaluation component, which evaluates all feedback coming from the tutor before it is delivered to the tutee. This evaluation is beneficial for both the tutor and tutee during their learning process in many ways. Firstly, the tutors can reflect on their own knowledge about the domain when their mistakes are notified by the system. During their tutoring process, if tutors receive the same type of tutee mistake on which they provided wrong feedback, the tutors will recall and try to avoid responding incorrectly. Secondly, the tutors have an opportunity to correct themselves before their feedback is delivered to the tutee, hence upholding their faith in teaching, and preventing them from thinking of themselves as inept tutors. Thirdly, the tutor's image in the tutee's perceptions is retained since the tutee always expects to receive the required feedback. Lastly, the tutee receives relevant feedback and models the solution without recording extra misconceptions in their log.

Figure 6 illustrates the overall problem-solving flow, and the roles of each student and the system as a whole. Bold parts in the flow diagram are related to tutor feedback evaluation. The feedback evaluation process starts when the tutor sends a response to the tutee student. The tutee student can take any one action: they can take step to design a solution (i.e., creating a class or modifying some properties, etc.), they can indicate that they have completed the solution by clicking the Evaluate button, or they can ask for the tutor's help by clicking the Hints button. The tutor can respond at any time after the tutee clicks the Evaluate button or Hints button. The main function of the feedback evaluator module is to evaluate the tutor feedback against the ideal solution and generate hints depending on the tutor's error. If the tutor has marked a correct mistake/error in the solution, then positive feedback from the system is generated to appreciate the tutor. In the opposite case, where the tutor marks a correct component as a mistake/error, negative feedback is displayed to the tutor indicating that the selected component is the correct one. Once the tutor marks the correct mistake/error, then the tutor needs to suggest the changes that are not present in tutee's solution. The feedback evaluator evaluates tutor's feedback regarding classes, attributes, methods, and relationships in the same ways as described above.

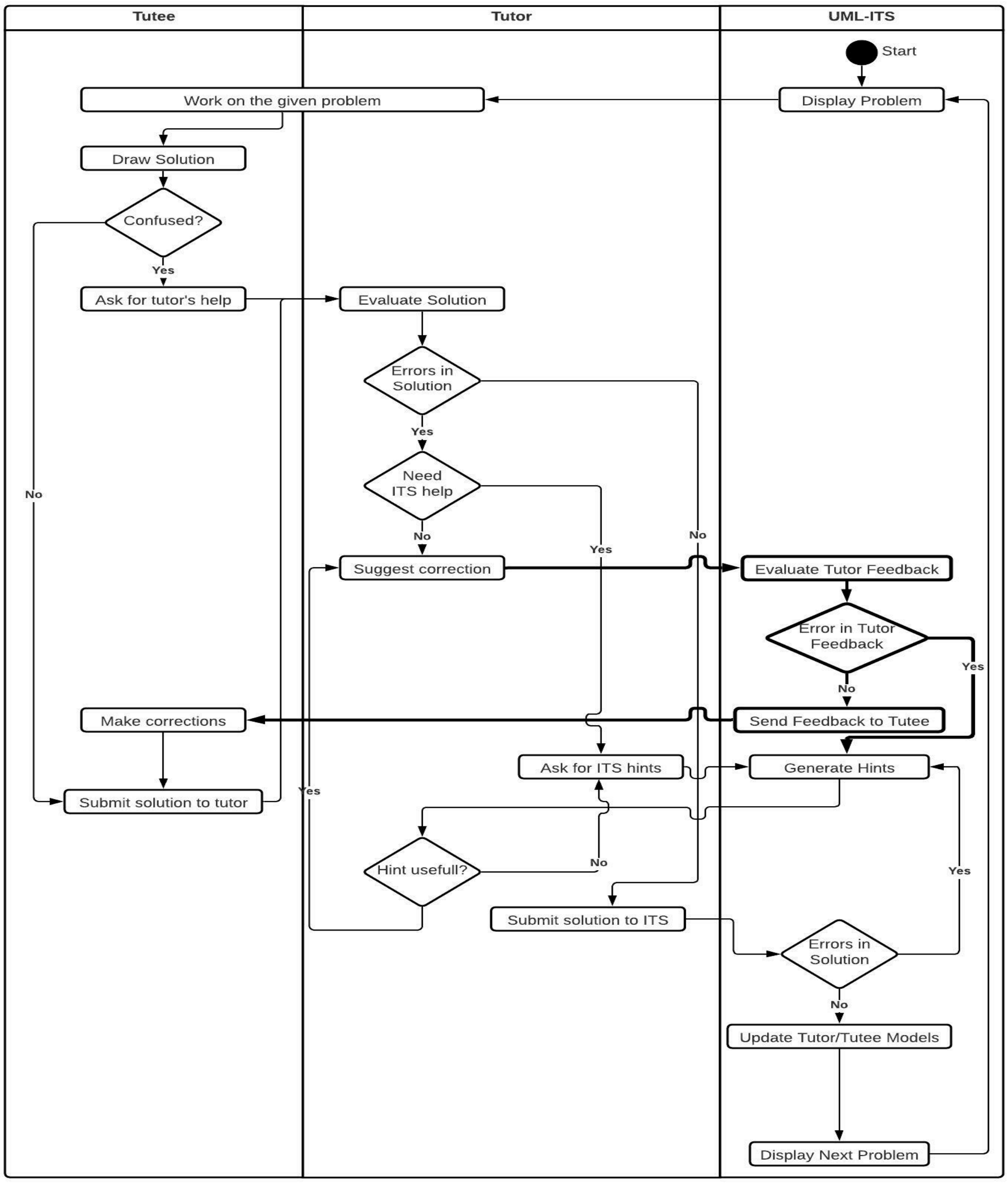


Figure 6. Swimlane diagram of the UML-ITS with peer feedback evaluation.

4. Evaluations

4.1. Experimental Design

To investigate our research questions and to determine the effects of the proposed model on a student's learning, an experimental study was conducted in which 100 students (57 female and 43 male) from different universities of Pakistan participated. All students were enrolled in different degree programs of Computer Science and they participated voluntarily. Fifty students used the UML-ITS without peer feedback evaluation (control group, in which tutors communicated erroneous feedback and suggestions to the tutee), and the other 50 students tutored each other using the UML-ITS with peer feedback evaluation (experimental group, in which tutors sent relevant and domain-related feedback to tutee students). The roles of tutor and tutee students were assigned randomly in both groups.

The study was conducted in two streams of three-hour laboratory sessions over two weeks, one week for each group. The students completed a pre-test (Supplementary S1) and then interacted with the UML-ITS, where each pair of tutor and tutee students worked on different UML class diagram scenarios. The students were seated in the same laboratory on different sides depending on their roles, and they were only allowed to communicate with one another via a chat tool provided with the system. After laboratory experimental sessions, students were asked to attempt a post-test (Supplementary S2), which was utilized to compare their results to their pre-test performance. The whole experimental design is depicted in Figure 7.

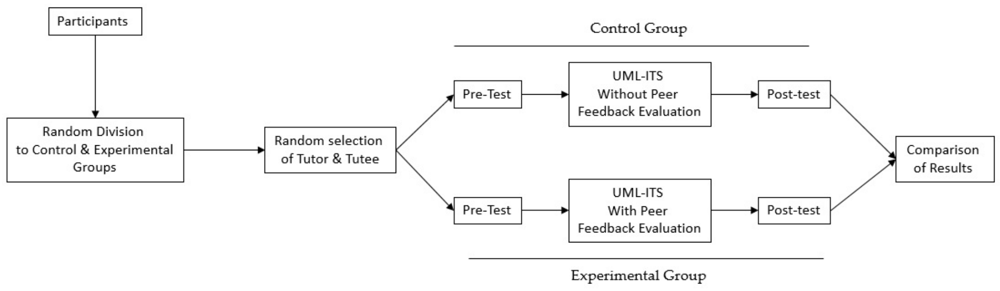

Participants
Random Division to Control & Experimental Groups
Random selection of Tutor & Tutee
Control Group
Pre-Test
UML-ITS Without Peer Feedback Evaluation
Post-test
Pre-Test
UML-ITS With Peer Feedback Evaluation
Post-test
Experimental Group
Comparison of Results

Figure 7. Experimental design.

4.2. Outcome Measures

In order to assess the performance of each student, pre- and post-tests were conducted. Each of these two tests contained a total of eight questions of 22 total marks. In the first question, students were asked to design a UML class diagram for the given problem scenario. In the second question, students were asked to write a description of the UML Class diagram. The remaining six questions were multiple-choice questions related to classes, attributes, and relationship types. All tests were administered on paper. The scores related to each question in both tests were calculated based on the number of correct class diagram components designed in the solution, the number of class attributes and relationships explained in the description, and the correct multiple-choice answers marked.

The findings of the pre- and post-tests were used to evaluate the students' performance. To monitor the difference between the pre-test and the post-test, the mean and standard deviation were calculated. Furthermore, the Normalized Learning Gain of each group was also calculated, which is the rough estimate of how efficient the prototype is at promoting the conceptual understanding of the subject. The formula presented in [36] was used and is shown in Equation (1):

$$\mathrm{NLG} = \frac{PostTestScore - PreTestScore}{100 - PreTestScore} \tag{1}$$

4.3. Problem-Solving Measures

To examine the student's problem-solving collaboration process in Control and Experimental groups, all actions taken by the tutee, tutor, and UML-ITS were recorded in log files. The following is the list of actions included in the log file:

- Number of problems solved by each group;
- Time taken to solve each problem;
- Whether each problem was successfully completed or not;
- Correct and incorrect problem-solving steps taken by each student;
- The number of times hints were requested or provided by UML-ITS.

5. Results

The assessment of the research hypothesis regarding peer feedback evaluation started with the comparison of pre-test and post-test results respective to both conditions, including the control group and experimental group. Then, log files that were generated during the experimental study were carefully analyzed to link individual student's actions with their own and their partner's learning gains. In short, the similarities and differences of domain learning pathways that students took in both conditions were made clear upon completion of this analysis.

5.1. The Effects of the Prototype on Student's Domain Learning

The most important measure of ITS effectiveness is the improvement in a student's domain knowledge. Table 3 contains absolute pre-test and post-test scores of students who participated in both conditions. It is worth noting that, despite student's prior domain familiarity, their pre-test scores were the lowest. To explore the differences between pre-test and post-test results across both conditions, an independent sample *t*-test and Mann–Whitney *z*-test were performed. It was observed that students in both groups showed improvement in their learning after prototype intervention and there was significant difference in pre-test and post-test results ($t = 5.067$, $p = 0.000$) ($z = 7.644$, $p = 0.000$). The effects of different conditions on students' learning were also investigated and their NLGs were compared with each other; see the last row of Table 3.

Table 3. Pre-test and post-test results.

Test Data	Mean	s.d.	*t*/*z*-Value	*p*-Value
		Control Group		
Pre-test	9.2	3.4	5.067	0.000
Post-test	13	4.0		
		Experimental Group		
Pre-test	9.1	4.3	7.644	0.000
Post-test	18.2	2.8		
		NLG Difference in both Conditions		
Control Group NLG	0.04	0.05	4.967	0.000
Experimental Group NLG	0.09	0.04		

Interestingly, there was significant difference between students' NLGs in both conditions ($z = 4.967$, $p = 0.000$). This reveals that the student's domain learning was affected by the conditions in which they were treated.

5.2. Total Number of Problem Scenarios Completed

To further investigate, the paths students took during the intervention were further explored by comparing the total number of questions or problem scenarios completed during the experimental session. It may be expected that each circumstance could have a similar problem-solving rate because of the significant difference between pre-test and post-test results, as explained in the above section. However, students in the control group attempted fewer problems (avg = 3.28) compared to students in the experimental group (avg: 7.26) (Table 4) since students in the control group had less relevant domain support.

Table 4. Total questions completed in two conditions.

Test Data	Mean	s.d.
Control Group	3.28	1.8
Experimental Group	7.26	1.29

In order to determine if problems completed were related to learning, the correlation of total problems successfully completed by each student with their normalized learning gain scores was calculated. Indeed, in the experimental group, problems successfully completed were marginally correlated with students' learning on NLG (r = 0.695, p = 0.000). However, the problems completed in the control group were also correlated with NLG results, but insignificantly (r = 0.176, p = 0.221), as shown in Table 5. Students in the control group received less domain support, due to which the rate of successful questions completed was lower than that of the other group. It can be inferred that if students attempt more problem scenarios, they have a chance to go through many concepts related to UML class diagrams. Those students who attempted fewer questions missed out some important concepts about which students were unclear, hence causing students to achieve a lower score in the post-test.

Table 5. Correlation—Total no. of questions completed vs. NLG.

Test Data	r	p
Control Group	0.176	0.221
Experimental Group	0.695	0.000

5.3. Problem-Solving Steps

In the above section, the difference between the total number of questions completed successfully in each condition was discussed. Although the questions completed in both conditions were correlated with student's NLG scores, students in the control group completed fewer problems compared to students of the other group. This is because the ITS did not provide them with the same support level. Furthermore, it can be hypothesized that students in the control group might appear to make more mistakes compared to those in the experimental group, again because of the lack of appropriate domain-level support from the ITS. To investigate this hypothesis, the total number of errors made by tutor and tutee students were compared in both conditions. According to the analysis results shown in Table 6, there was significant difference between tutees' errors made in both conditions (t = 7.798, p = 0.003), indicating that tutees in the control group made more errors during experimental intervention compared to tutee students in the experimental group. Interestingly, the tutor students in both conditions made identical errors as there is no significant difference (t = 1.440, p = 0.157), despite providing more domain-related help in the experimental group.

Table 6. No. of tutee and tutor errors in both conditions.

Test Data	Statistical Test	t	p
No. of Tutee errors	Independent Sample t test	7.798	0.003
No. of Tutor errors	Independent Sample t test	1.440	0.157

To investigate the effects of tutors' errors on the tutees' learning path, as hypothesized in previous sections, the correlation between the total number of errors made by the tutees was compared with the total number of errors made by the tutors in both conditions. As shown in Table 7, the errors made by the tutors in the control group were positively correlated with the total number of errors made by the tutees. This is because the tutors' wrong suggestions/mistakes were given to the tutees and the tutees followed those erroneous suggestions, which resulted in the higher rate of the tutees' errors. The errors made by the tutors in the experimental group were negatively correlated with the total number of errors made by the tutees.

Table 7. Correlation—total number of tutor and tutee errors in both conditions.

	Control Group		**Experimental Group**	
Test Data	*r*	*p*	*r*	*p*
No. of Tutor errors vs. Tutee errors	0.469	0.018	−0.060	0.776
No. of errors as Tutee vs. NLG	−0.270	0.191	−0.083	0.693
No. of errors viewed as Tutor vs. NLG	−0.289	0.161	0.362	0.010

It was also found that if the tutees made errors and they were viewed by the tutors, it was related to the tutors' learning in both conditions. The total number of errors made by the tutees in both conditions was negatively correlated to NLG. The total number of errors viewed by the tutors in the control group was also negatively correlated to the student's NLG. However, the viewing errors of the tutors in the experimental group were positively correlated to their NLG. It appeared that tutors who observed their tutees' inability to progress (when tutees made mistakes) was in fact connected to learning from tutoring. The overall correlation results imply that tutor students are indeed taking advantage of the ITS's peer feedback evaluation feature to reflect on their erroneous suggestions and rectifying them before sending then to the tutee.

5.4. *Effects of Collaboration and Peer Feedback*

At this level, the interaction between tutors, tutees, and the Intelligent Tutoring System was investigated. The first step in this analysis was to determine the effects of tutees' help-seeking behavior on their learning gains. It is believed that the students who ask for help when it is needed tend to learn more. As explained earlier, the errors viewed by tutors were correlated with their learning, but errors made by tutees were negatively correlated to their learning gains. To further explore the elements that affected tutees' learning, hints requested, correct feedback received, and incorrect feedback received were correlated with tutees' learning gains (Table 8).

Table 8. Correlation—collaboration and peer feedback.

	Control Group		**Experimental Group**	
Test Data	*r*	*p*	*r*	*p*
Hints Requested	0.490	0.00	0.386	0.003
Correct Feedback Received	0.558	0.000	0.783	0.000
Incorrect Feedback Received	−0.460	0.000	0.201	0.161
Hint Request Received	0.505	0.000	0.781	0.000
Incorrect Feedback Sent	−0.392	0.002	0.669	0.000

It can be observed that hints requested in both conditions were correlated to tutees' learning gains (r = 0.490, p = 0.000) (r = 0.386, p = 0.003). Correct feedback received by tutees in both conditions was also correlated to tutees' learning gains (r = 0.558, p = 0.000) (r = 0.783, p = 0.000). This indicates that correct feedback from tutors on tutees' solutions do have a positive effect on tutees' domain learning. Moreover, the incorrect feedback received from tutors in both conditions did not contribute to tutees' learning gains, which addresses our research question.

The next stage was to determine the factors that influenced the tutors' learning gains, because if tutees' hint-taking is linked to their learning, it is likely that receiving hint requests as a tutor will also contribute to their domain learning. Receiving hint requests as a tutor in both conditions was positively correlated to the tutor's learning gain (r = 0.505, p = 0.000) (r = 0.781, p = 0.000). It can be inferred that tutors learn more when they receive more hint requests from a tutee, because when tutors receive hints, they try to overcome the errors present in the tutee's solutions; hence, receiving hint requests encourages them to study more about the solutions.

Taking into account the opposite side of the story, when tutors receive hint requests, their learning improves. This depends on whether they send domain-related feedback that is in accordance with the solution, i.e., the feedback contains accurate information about the tutee's errors/mistakes in the diagram, or they send incorrect responses based on incorrect assumptions about the tutee's solutions. To explore this, incorrect feedback sent was correlated to tutors' learning gains. As shown in Table 8, the incorrect feedback provided by tutors in the control group to tutees was negatively correlated to their learning gains ($r = -0392$, $p = 0.005$). Here, because the tutors did not receive domain-level help from the UML-ITS, it is probable that they misinterpreted the tutee's solutions and replied with erroneous suggestions, resulting in a reduction in the tutee's learning gains. On the other hand, the tutor's incorrect feedback was correlated to their learning gains in the experimental group ($r = 0.669$, $p = 0.000$). The UML-ITS prevented tutors from sending incorrect feedback and provided them with domain level hints so that tutors could rethink the suggestions they were trying to send. Receiving domain level hints from the system allowed tutors to reflect on their own learning first and then send correct responses to the tutee. Furthermore, as previously mentioned, correct feedback received by tutees from a tutor was also linked to their learning improvements. In conclusion, both tutor and tutee students benefited from the UML-ITS by sending and receiving proper feedback on solutions.

5.5. Regression Analysis

As a last step, regression analysis was carried out to evaluate the abilities of the variables described in the previous sections to predict students' learning gains. The six factors of the students as tutors and tutees were considered to build a model for domain learning prediction in both conditions (control and experimental groups). The model contains the total number of questions completed, hints requested, correct feedback received, incorrect feedback received, hints requests received, and incorrect feedback sent. The model explained about 51% of the variation in learning gain as a whole ($R^2 = 0.511$, $F = 9.548$, $p = 0.000$) in the control group (Table 9). Of the six variables, three significantly predicted students' leaning gains, correct feedback received ($\beta = 0.415$, $t = 3.497$, $p = 0.001$), hint requests received ($\beta = 0.387$, $t = 2.119$, $p = 0.040$), and number of questions completed ($\beta = 0.252$, $t = 2.496$, $p = 0.016$). The remaining variables were either negatively predicted by the model or did not provide a significant prediction. While the three variables were significantly correlated to learning gains in control group, it appears that receiving a correct domain-related response from a tutor helped tutee students to overcome their learning gaps. On the other hand, receiving hint requests from a tutee also encouraged tutors to locate and correct errors in tutees' solutions, hence predicting the learning gains. It also appears that students receiving or sending incorrect feedback had a negative impact on their learning gains which, in this case, was due to the lack of appropriate system domain-level support.

Table 9. Regression analysis to predict student's learning gains.

Variables	**Control Grout**			**Experimental Group**		
	β	t	p	β	t	p
Questions Completed	0.252	2.496	0.016	0.186	2.086	0.043
Hints Requested	−0.055	−0.286	0.776	−0.147	−1.666	0.103
Correct Feedback Received	0.415	3.497	0.001	0.478	4.245	0.000
Incorrect Feedback Received	−0.022	−0.170	0.866	−0.070	−1.016	0.315
Hint Request Received	0.387	2.119	0.040	0.298	3.043	0.004
Incorrect Feedback Sent	−0.296	−2.634	0.012	0.241	2.920	0.006

Another type of regression analysis was conducted to predict the learning gains of the experimental group. The model contains the same six variables that were used to predict the learning gains of the control group, including the total number of questions completed, hints requested, correct feedback received, incorrect feedback received, hints requests received, and incorrect feedback sent. A significant percentage of the variance in the learning gain was explained by the model (R^2 = 0.799, F = 33.557, p = 0.000) (Table 9), although due to the small sample size it is likely that this value is inflated [37].

It can be observed from the regression analysis of the experimental group that two variables that do not significantly predict the gain in students' learning: hints requested ($\beta = -0.147$, $t = -1.666$, p = 0.103) and incorrect feedback received ($\beta = -0.070$, $t = -1.016$, p = 0.315). Interestingly, hints requested in both models does not predict students' learning gains, although, as mentioned in the previous section, hints made was positively correlated to students' learning gains. In general, this is because students may have requested help when it actually was not needed or before drawing components on the workspace. On the other hand, the tutor students who received hint requests in that way indeed provided wrong feedback. In this case, the wrong feedback was recorded when tutors suggested something through chat conversations. Apart from these two factors, every other variable substantially predicted students' learning gains, showing that the dual roles of tutor and tutee benefitted the students.

6. Discussion

This study proposed an ITS design with peer feedback evaluation and investigated its usage effects on students' learning. It was hypothesized that tutee students who receive erroneous feedback from their peer tutors would have poorer learning and overall performance than those who receive correct and domain-related feedback. It can be observed from the findings that students in both groups had improvements in their post-test results, but there was a significant difference in their learning gains in each group. This was due to the paths students took during the experimental period and certain design elements that had unique effects on students' learning in both conditions. The UML-ITS, for example, did not support tutors in the control group when they were requested to assist tutees. As a result, the tutor students gave erroneous hints, which UML-ITS evaluated when tutee students included them in their solutions. Under such situations, tutor and tutee students made more mistakes and scored less in their post-test as compared to the other group. On the other hand, students in the experimental group showed a significant improvement in their post-tests results because of receiving and sending domain-related feedback. Again, this was due to the proper domain level support from the UML-ITS. When tutors were not able to locate the mistakes in tutees' solutions or provided incorrect feedback, the UML-ITS generated hints, after evaluating the tutors' feedback, which helped tutors to revise the solution and go through it again. Furthermore, tutee students also received correct solution-related feedback from the tutors.

After the evaluation study and outcomes, it was possible to respond to the research question addressed by this paper. Some evidence was discovered in this study that suggested that receiving wrong responses had a detrimental influence on students' learning gains. For example, if tutee students followed tutors' wrong suggestions, they experienced a significant increase in their errors made (tutees' errors plus tutors' wrong suggestions that were implemented by the tutees in solutions). Not only did their number of total errors made increased, but students in those groups were also not able to attempt more problem scenarios compared to those who received domain-related responses from tutors. The tutors who were not notified about their incorrect feedback, on the other hand, also did not have the opportunity to reflect on their suggestions. Furthermore, attempting fewer problems, making more mistakes, and sending/receiving incorrect responses were all not correlated to their learning gains. Hence, receiving incorrect responses from tutors has a negative impact on tutees' learning gains. Nevertheless, based on the findings of this study, it seems that the benefits of peer feedback evaluation will grow as its quality improves.

7. Conclusions

Intelligent Tutoring Systems are computerized systems that help students in learning different subjects. These systems are gaining popularity due to the fact that they are available all the time and are easy to access and use. This paper presented UML-ITS, an intelligent tutoring for teaching UML with a peer feedback evaluation component. The empirical study included control and experimental groups (with and without peer feedback evaluation) to determine the effects of the ITS model on students' domain learning. While teaching the design of UML class diagrams, the experimental group also received support from the UML-ITS to evaluate peer feedback for its correctness and relevancy against an ideal solution. The system's peer feedback evaluation component double-checks all feedback from tutors before delivering it to the tutee student, which not only improved tutees' learning skill, but also helped peer tutors to rethink their own solutions, indicating a better influence on learning from both sides. In short, the students in both conditions showed an improvement in their domain knowledge, but students with peer feedback evaluation performed significantly better on their post-test after UML-ITS session, indicating that they gained greater expertise in UML modeling. Hence, it can be concluded that peer feedback evaluation in the ITS appears to be a promising advancement and should be implemented with enhancements in future ITS tools.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/computers11030030/s1, Supplementary S1: Pre-Test; Supplementary S2: Post-Test.

Author Contributions: Conceptualization, methodology, S.A., H.K. and A.B. (Amber Baig); software, S.A., M.U.R. and A.B. (Amber Baig); validation, formal analysis, investigation, resources, H.K., M.U.R. and A.B. (Ahsanullah Baloch); writing—original draft preparation, S.A., M.U.R. and A.B. (Amber Baig); writing—review and editing, visualization, H.K., M.U.R. and A.B. (Ahsanullah Baloch); supervision, H.K., M.U.R. and A.B. (Ahsanullah Baloch). All authors have read and agreed to the published version of the manuscript.

Funding: Research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baker, R.S.; D'Mello, S.K.; Rodrigo, M.M.T.; Graesser, A.C. Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive–affective states during interactions with three different computer-based learning environments. *Int. J. Hum.-Comput. Stud.* **2010**, *68*, 223–241. [CrossRef]
2. Chrysafiadi, K.; Virvou, M. Student modeling approaches: A literature review for the last decade. *Expert Syst. Appl.* **2013**, *40*, 4715–4729. [CrossRef]
3. VanLehn, K. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educ. Psychol.* **2011**, *46*, 197–221. [CrossRef]
4. Ma, W.; Adesope, O.O.; Nesbit, J.C.; Liu, Q. Intelligent tutoring systems and learning outcomes: A meta-analysis. *J. Educ. Psychol.* **2014**, *106*, 901. [CrossRef]
5. Woolf, B.P. *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing e-Learning*; Morgan Kaufmann: Burlington, MA, USA, 2010.
6. Graesser, A.C. Conversations with AutoTutor help students learn. *Int. J. Artif. Intell. Educ.* **2016**, *26*, 124–132. [CrossRef]
7. Graesser, A.C.; Dowell, N.; Hampton, A.J.; Lippert, A.M.; Li, H.; Shaffer, D.W. Building intelligent conversational tutors and mentors for team collaborative problem solving: Guidance from the 2015 Program for International Student Assessment. In *Building Intelligent Tutoring Systems for Teams*; Emerald Publishing Limited: Bingley, UK, 2018.
8. Katz, S.; Aronis, J.; Creitz, C. Modeling pedagogical interactions with machine learning. *Kognitionswissenschaft* **2000**, *9*, 45–49. [CrossRef]

9. Tahir, F.; Mitrovic, A.; Sotardi, V. Investigating the effects of gamifying SQL-Tutor. In Proceedings of the 28th International Conference on Computers in Education, Virtual, 23–27 November 2020; Asia-Pacific Society for Computers in Education: Taiwan, 2020.
10. Baghaei, N.; Mitrovic, A. A Constraint-Based Collaborative Environment for Learning UML Class Diagrams. In Proceedings of the International Conference on Intelligent Tutoring Systems, Jhongli, Taiwan, 26–30 June 2006; pp. 176–186.
11. Baghaei, N.; Mitrovic, A. Evaluating a collaborative constraint-based tutor for UML class diagrams. In Proceedings of the 13th International Conference on Artificial Intelligence in Education, Los Angeles, CA, USA, 9–13 July 2007; pp. 533–535.
12. Holland, J.; Baghaei, N.; Mathews, M.; Mitrovic, A. The effects of domain and collaboration feedback on learning in a collaborative intelligent tutoring system. In *International Conference on Artificial Intelligence in Education*; Springer: Berlin, Heidelberg, 2011; pp. 469–471.
13. Eid, M.I. A learning system for entity relationship modeling. In Proceedings of the PACIS 2012 Proceedings, Paper 152, Ho Chi Minh City, Vietnam, 11–15 July 2012.
14. Suraweera, P.; Mitrovic, A. KERMIT: A Constraint-based Tutor for Database Modeling. In Proceedings of the 6th International Conference on Intelligent Tutoring Systems 2002, San Sebastian, Spain, 2–7 June 2002; pp. 377–387.
15. Suraweera, P.; Mitrovic, A. An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artif. Intell. Educ.* **2004**, *14*, 375–417.
16. Tan, P.J. Applying the UTAUT to understand factors affecting the use of English e-learning websites in Taiwan. *Sage Open* **2013**, *3*, 2158244013503837. [CrossRef]
17. Liu, L.; Chen, L.; Shi, C.; Chen, H. The Study of Collaborative Learning Grouping Strategy in Intelligent Tutoring System. In Proceedings of the 14th International Conference on Computer Supported Cooperative Work in Design CSCWD 2010, Shanghai, China, 14–16 April 2010; pp. 642–646.
18. Narciss, S. Feedback strategies for interactive learning tasks. In *Handbook of Research on Educational Communications and Technology*; Spector, J.M., Merrill, M.D., van Merrieboer, J., Driscoll, M.P., Eds.; Taylor & Francis Group: New York, NY, USA, 2010; pp. 125–143.
19. Gielen, S.; Peeters, E.; Dochy, F.; Onghena, P.; Struyven, K. Improving the effectiveness of peer feedback for learning. *Int. J. Learn. Instr.* **2010**, *20*, 304–315. [CrossRef]
20. Jahin, J.H. The effect of peer reviewing on writing apprehension and essay writing ability of prospective EFL teachers. *Aust. J. Teach. Educ.* **2012**, *37*, 65–89. [CrossRef]
21. Wankiiri-Hale, C.; Maloney, C.; Seger, N.; Horvath, Z. Assessment of a student peer-tutoring program focusing on the benefits to the tutors. *J. Dent. Educ.* **2020**, *84*, 695–703. [CrossRef]
22. Dioso-Henson, L. The effect of reciprocal peer tutoring and non-reciprocal peer tutoring on the performance of students in college physics. *Res. Educ.* **2012**, *87*, 34–49. [CrossRef]
23. Evans, M.J.; Moore, J.S. Peer tutoring with the aid of the Internet. *Br. J. Educ. Technol.* **2013**, *44*, 144–155. [CrossRef]
24. Worley, J.; Naresh, N. Heterogeneous peer-tutoring: An intervention that fosters collaborations and empowers learners: Key features of an intervention peer-tutoring program highlight the cognitive and social benefits of this collaborative approach. *Middle Sch. J.* **2014**, *46*, 26–32. [CrossRef]
25. Alegre, F.; Moliner, L.; Maroto, A.; Lorenzo-Valentin, G. Peer tutoring in algebra: A study in middle school. *J. Educ. Res.* **2019**, *112*, 693–699. [CrossRef]
26. Alegre-Ansuategui, F.J.; Moliner, L.; Lorenzo, G.; Maroto, A. Peer tutoring and academic achievement in mathematics: A meta-analysis. *Eurasia J. Math. Sci. Technol. Educ.* **2018**, *14*, 337–354. [CrossRef]
27. Leung, K.C. Compare the moderator for pre-test-posttest design in peer tutoring with treatment-control/comparison design. *Eur. J. Psychol. Educ.* **2019**, *34*, 685–703. [CrossRef]
28. Hardavella, G.; Aamli-Gaagnat, A.; Saad, N.; Rousalova, I.; Sreter, K.B. How to give and receive feedback effectively. *Breathe* **2017**, *13*, 327–333. [CrossRef]
29. Harsley, R.; Green, N.E.; di Eugenio, B.; Aditya, S.; Fossati, D.; Al Zoubi, O. Collab-ChiQat: A Collaborative Remaking of a Computer Science Intelligent Tutoring System. In Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion—CSCW '16 Companion, San Francisco, CA, USA, 26 February–2 March 2016.
30. Duangnamol, T.; Suntisrivarporn, B.; Supnithi, T.; Ikeda, M. Circuitously Collaborative Learning Environment to Enhance Metacognition. In Proceedings of the International Conference on Computers in Education, Nara, Japan, 30 November–4 December 2014; Asia-Pacific Society for Computers in Education: Nara, Japan, 2014; pp. 1–4.
31. Haq, I.U.; Anwar, A.; Basharat, I.; Sultan, K. Intelligent Tutoring Supported Collaborative Learning (ITSCL): A Hybrid Framework. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 523–535. [CrossRef]
32. Sychev, O.; Penskoy, N.; Anikin, A.; Denisov, M.; Prokudin, A. Improving Comprehension: Intelligent Tutoring System Explaining the Domain Rules When Students Break Them. *Educ. Sci.* **2021**, *11*, 179. [CrossRef]
33. Polito, G.; Temperini, M. A gamified web based system for computer programming learning. *Comput. Educ. Artif. Intell.* **2021**, *2*, 100029. [CrossRef]
34. Kumar, A.N. Allowing Revisions While Providing Error-Flagging Support: Is More Better? In Proceedings of the 21st International Conference on Artificial Intelligence in Education, Ifrane, Morocco, 6–10 July 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 147–151.

35. Kumar, A.N. Limiting the Number of Revisions while Providing Error-Flagging Support during Tests. In Proceedings of the 11th International Conference on Intelligent Tutoring Systems, Chania, Greece, 14–18 June 2012; Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 524–530.
36. Abbasi, S.; Kazi, H.; Kazi, A.W.; Khowaja, K.; Baloch, A. Gauge Object Oriented Programming in Student's Learning Performance, Normalized Learning Gains and Perceived Motivation with Serious Games. *Information* **2021**, *12*, 101. [CrossRef]
37. Whitehead, A.L.; Julious, S.A.; Cooper, C.L.; Campbell, M.J. Estimating the sample size for a pilot randomized trial to minimize the overall trial sample size for the external pilot and main trial for a continuous outcome variable. *Stat. Methods Med. Res.* **2016**, *25*, 1057–1073. [CrossRef] [PubMed]