Improving Learning Outcomes in UML Sequence Diagrams Through Reduced Cognitive Load

Sohail Alhazmi
RMIT University
Melbourne, Australia *sohail.alhazmi@rmit.edu.au*

Charles Thevathayan
RMIT University
Melbourne, Australia *charles.thevathayan@rmit.edu.au*

Margaret Hamilton
RMIT University
Melbourne, Australia *margaret.hamilton@rmit.edu.au*

Abstract

This paper demonstrates how cognitive load theory can be used to improve learning outcomes by presenting a tool capable of assisting novices to learn to model sequence diagrams effectively. Sequence diagrams are known to lead to heavy cognitive load as they must be consistent with the class diagram, while discharging all the responsibilities specified in the underlying use case. Moreover, novices must also consider the various design options and their impact on the qualitative aspects of the model.

Our tool allows cognitive load to be better managed by using a 'divide and conquer' approach. In the initial stage students need to focus only on consistency aspects, and they will not be allowed violate the constraints stated in the class diagram. In the second stage, students will not be allowed to submit a diagram until the stated use case goals are met. In the final stage qualitative feedback and marks are awarded based on established metrics and students are allowed to improve their scores by resubmitting the model. Qualitative and quantitative results show that our novel tool using a form of gamification has helped to improve the learning outcomes in modelling substantially, especially for the stragglers. One benefit of our approach is that it can be adapted to other areas where students maybe cognitively challenged.

Keywords: Cognitive Load Theory, Interaction Diagrams, e-learning Tool, UML Modelling, Sequence Diagrams

1. Introduction

One of the key responsibilities of a systems analyst is to develop analysis models that identify the architecture of an information system**.** The Unified Modelling Language (UML) is the most common notation used to describe these models. UML has evolved into a standard object-oriented (OO) modelling notation for designing conceptual models of software systems [21]. Students are exposed to UML early in their study as it plays a crucial role in specifying, visualizing, constructing, and documenting OO software system [15]. Students use UML models to validate design decisions and to examine their influence on quality. With UML as the standard modelling notation in industry, proficiency in UML is a valuable asset for every information system novice.

Developing good modelling skills are cognitively demanding and students are expected to acquire them as they progress through various courses. Despite design and UML being a core element in IS and CS degrees, a previous multi-institutional study with 314 participants found that the majority of graduating students were unable to design a software system [11]. A number of early studies investigating student performance in modelling

revealed students face the greatest difficulties in modelling interaction diagrams [4], [23], affirming our own experiences working with novices in software engineering (SE) projects. The problem is exacerbated for sequence diagrams (SDs), as the high number of interacting items that must be handled concurrently in working memory greatly increases the cognitive load [24].

For learning to be effective, formative feedback on heavy cognitive tasks such as modelling become necessary for good learning outcomes in subsequent years [22]. With diverse students, ongoing diagnosis is needed for each individual student facing difficulties. However, educators teaching software engineering modelling are facing many challenges as massive class sizes and post-COVID classes make meaningful one-to-one interaction and feedback in the early formative stages difficult. Providing timely feedback to increasingly diverse student cohorts therefore relies on developing learning tools able to guide and provide immediate feedback when students are learning to model.

The cognitive load theory (CLT) postulates that the cognitive load resulting from a task may potentially hamper learning [27]. Any strategy that involves more cognitive load than available working memory can deteriorate performance by overwhelming the learner [29]. Modelling sequence diagram overwhelms many learners as it involves a high number of interacting items that must be concurrently handled in the working memory such as the messages that can initiate other messages, the objects that can receive the messages and the type of messages that can be sent [29]. The messages that can initiate other messages are determined by whether it is one of those in the current method stack. The object to which a message can be directed depends on whether there is an association, parameter or local link. If there is a one-to-many or many-to-many associations in the class diagram a reference to the relevant object must be obtained usually by making a self-call, but class diagrams may not capture such internal details. The type of elements that can be sent as part of the message is determined based on its method signature in the class diagram.

Many have asserted the need to develop techniques to reduce the high cognitive load required for learning sequence diagrams [25], [29], which stems from the need to consider constraints and requirements imposed by multiple other models, qualitative aspects and the current state of the model. Research has shown cognitive load during modelling can be lowered by subdividing the problem into meaningful parts thereby reducing the number of interacting elements that must be considered concurrently [29]. However, none of the educational modelling tools surveyed have the ability to diagnose and nudge students towards a valid solution or to reduce the cognitive load. To the best of our knowledge, no pedagogy-based tool has been devised for teaching modelling SDs, though many novices are known to face severe difficulties in learning this multifaceted and important modelling artifact.

These problems have led to the research question which we address in this paper:

- How can an e-learning tool be devised to reduce the high cognitive load many students face when learning to model sequence diagrams?

The rest of the sections are described as follows. Section 2 presents the related work of this study. Section 3 details our methodology. We briefly explain the proposed tool in Section 4. Section 5 presents the results which are followed by discussion, limitations and conclusion.

2. Related Work

2.1. Cognitive Load Theory (CLT)

The vast majority of research on CLT is concerned with reducing the cognitive load by managing the design of instructional materials [14], [18], known as the extraneous load. Research has identified several effects that can reduce the extraneous load. Some of these include goal-free effect, worked example effect, completion problem effect, split-attention effect, modality effect, and redundancy effect [28], [30], most of which are intended to deal with the presentation of instructions. Since the UML standard dictates the presentation of the sequence diagram, the extraneous load is considered irrelevant.

Recent studies have attempted to reduce the cognitive load inherent to the task by

reducing the number of interacting elements involved in a task [8], [30], known as the intrinsic load. One method of reducing intrinsic load is to subdivide the complex task into meaningful modules that can be handled and learned separately [12]. Since the working memory treats each part as a single unit of information, the modular approach can significantly reduce the cognitive load. It is possible to follow this modular approach in SD modelling because some activities such as modelling SDs consistent with class diagrams can be handled separately from other activities. Similarly, identifying the final goal (completeness) for each SD model can be defined by interpreting all the responsibilities specified in the underlying use case. This essentially reduces the number of interacting elements involved in SD modelling; therefore, reducing the intrinsic load. Reducing the intrinsic load will free up the working memory and cognitive resource.

A few studies have emphasized the need to develop approaches to reduce the high cognitive load required for learning SDs [25], [29], which stems from the need to consider constraints and requirements imposed by multiple other models, qualitative aspects and the current state of the model. Research has shown cognitive load during modelling can be lowered by subdividing the problem into meaningful parts thereby reducing the number of interacting elements that must be considered concurrently [29].

2.2. Existing UML Modelling Tools

Several UML modelling tools were designed in the past for various demands and user profiles with varying levels of complexity. Most of these were intended primarily at practising engineers [19]. Such tools with many advanced characteristics and functionalities (e.g. forward or reverse engineering) aided company designers to model large complex systems. Visual tools have been developed for multi-domain modeling (e.g. computational modeling) including MagicDraw, Modelio2, UMLet, QuickUML and minimUML2 [1]. Several studies have highlighted problems when industrial tools are used in education [1, 2], [20]. Being designed for experts they are often unduly complex, difficult to learn, and distracting for students. In addition, these tools do not provide any useful feedback on the design aspects. Moreover, these tools often do not consider the novices' difficulties and challenges that can alleviate the cognitive loads on novices' modelers [20]. Therefore, some of the important educational aspects such as consistency checking, inter-model dependency and ongoing diagnoses are absent in industrial tools as these tools targeted to experts and not for novices.

In the recent past, a few non-commercial UML tools have been designed specifically for the educational context [5, 6], [10], [16]. In our previous study [3], we surveyed tools developed for educational context and summarized the strengths and weaknesses of the existing educational UML tools. The results showed that, 1) no past attempts to provide instant feedback on design tasks considering CLT to reduce high cognitive load in modelling task, 2) none of the tools surveyed were able to give qualitative feedback using metrics such as coupling and cohesion, and 3) none of the educational modelling tools surveyed have the ability to diagnose and nudge students towards a valid solution or to reduce the cognitive load.

Employers are increasingly demanding graduates come with Higher Order Thinking Skills (HOTS) with inherently high cognitive load, while institutions are relying more on online teaching. We therefore posit, more research must be undertaken to effectively facilitate CLT infused e-learning techniques.

3. Methodology

The purpose of this study was to reduce the cognitive load for novices learning modelling SDs by developing an e-learning tool based on the CLT recommendations to manage the cognitive load complexity of modelling SDs.

To evaluate the effectiveness of the e-learning tool, multiple sources of data were collected, including pre-tests and post-tests, a survey gathering the effectiveness of the tool from students' perspectives and data collected through the tool (such as the number of

attempts and the time taken). The Software Engineering Fundamentals (SEF) course where this e-learning tool was trialed is a core course for all students taking a degree in Computer Science, IT, IS or Software Engineering at our University both at post-graduate (PG) under-graduate (UG) levels. The SEF enrollment varies from 100 to 300 students each semester with increasingly diverse cohorts. However, the number of students have dramatically declined due to COVID'19. The SEF course covers many abstract software engineering concepts, including UML design, patterns and principles. Most incoming students having little or no prior experience with object-oriented programming skills and find UML design difficult and abstract. Each student was asked to undertake a pre-test before interacting with the tool, followed by a post-test and a survey.

Pre- and post-tests were designed to measure the performance 'improvement' after using the new tool. These questions were designed to cover varies levels of the Bloom's taxonomy [9] up to analysis level. These tests focused on UML design concepts, specifically SDs as these were the main learning objectives for our proposed approach. Some questions are designed mainly to test whether students have understood; 1) the interdependencies between domain models (class diagram) and interaction models (sequence diagrams), 2) the impact of post-conditions stated in use case on modelling SD, and 3) the importance of qualitative aspects for modelling optimal quality of SDs. Figure 1 shows a sample question from the pre/post-test question bank designed to measure whether students have understood the interdependencies between the class diagram and SD after using the tool.

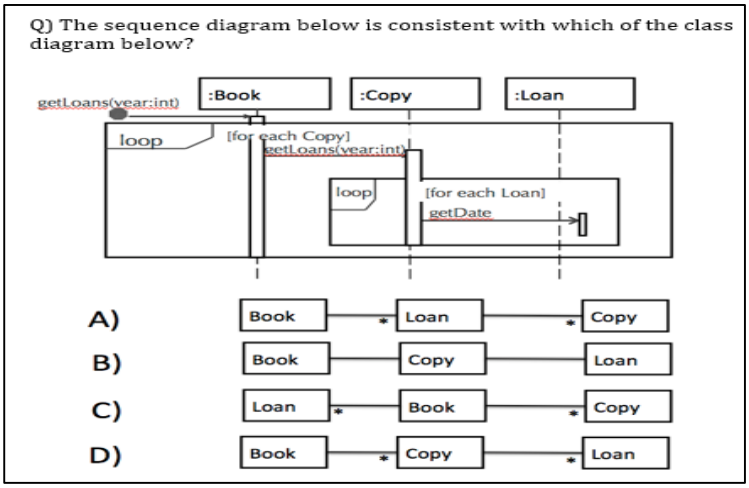

Figure 1. A sample pre/post-test question designed to measure student understanding of SD concepts.

Statistical techniques were used to measure whether the tool helped to improve learning outcomes among our diverse student cohort. Multiple use-case scenarios of varying difficulty levels were presented for students in the tool. We have analyzed the pre- and post- test questions that related to the different aspects of SDs. The improvement in performance was determined based on the difference between pre- and post-test scores.

Student perceptions were measured using students' feedback related to the tool through a specially designed survey. A questionnaire was designed to identify how the feedback can be improved in the subsequent iteration and how they perceived the effectiveness of the tool in improving their design skills. We designed a range of Likert-scale type questions and open-ended questions.

3.1. Managing Cognitive Complexity of Sequence Diagrams Modelling In the New E-Learning Modelling Tool

In our previous work [3], we have conducted an exploratory study to identify the most common difficulties facing novice modelling SDs. Several main difficulties were identified and classified as root causes of novices' difficulties with sequence diagrams. The following Table shows a summary of our findings for novice modelling difficulties:

Table 1. Difficulties with Modelling Sequence Diagrams.

	Identified Novice' Difficulties	Difficulty Classification
1	Students do not realize messages and message elements dispatched in sequence diagrams must be consistent with class diagrams.	Consistency verification
2	Students have difficulty detecting when a sequence-diagram is complete as post-conditions stated in use cases can be difficult to interpret.	Poorly defined goals. (completeness verification)
3	When deciding which message can be dispatched, students often fail to consider parameter links, local links and links returned earlier as well as the attributes returned by the local and external methods.	Difficulty tracking current knowledge state
4	Poor quality of designs mostly resulting in highly coupled and centralized designs.	Lack of feedback on qualitative aspects

These difficulties suggest high intrinsic load as the main reason why novices were overwhelmed by modelling SDs. Students had to track the current state while ensuring consistency with class diagrams, meet goals stated in use cases, adhere to implicit constraints inherent in the domain as well as come up with optimal design considering qualitative aspects. These challenges led us to find novel ways to reduce the cognitive load in active memory by decomposing the modelling tasks.

Stage 1: Managing the Most Fundamental Difficulty (Model Consistency Verification)

Our tool provided a way to structure the cognitive loads to suit diverse students by requiring novices to focus firstly only on consistency and constraints aspects (most fundamental pressing problem that faced novices modelling SDs) through the instant feedback. Our tools developed ensure messages in SDs are consistent with domain models by allowing only valid messages to be dispatched at any stage by aggregating all past interactions. For each message, the student specifies the target object as well as the list of parameters. If the message is valid the system updates the state of interaction, visually depicts the message and allows the student to proceed, otherwise appropriate diagnostic messages are displayed to guide the novices. Consistency type checking against the class diagram is performed based on the class of the message target object and the type of message elements. In addition, for a message to be valid, the knowledge elements at the source must be a superset of the message elements dispatched and must contain a link to the target object. Figure 2 shows an example of an instant error feedback provided after adding an invalid message.

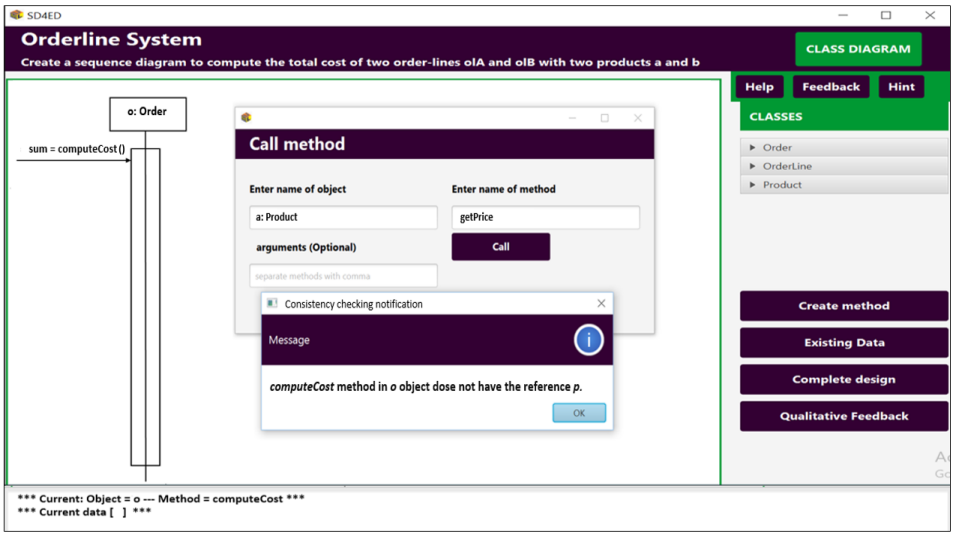

Figure 2. Example of instant error message generated by tool if consistency constraints are violated.

Stage 2: Model Completeness Verification

In the second stage, when students are aware of how SDs are constrained by the underlying class diagram, students will not be allowed to submit a diagram until the stated use case

goals are met. So, novices are provided with pre-created objects whose references are explicitly captured in the tool. Students need to meet all the use case post-conditions. Hence the system knows all objects and can provide feedback when some post-conditions are not met. Figure 3 shows an example of the feedback provided when checking the model meets all the post-conditions and students can view the details of post-conditions yet to be met. Hint buttons are incorporated to allow students to step through towards the goal state one step at a time.

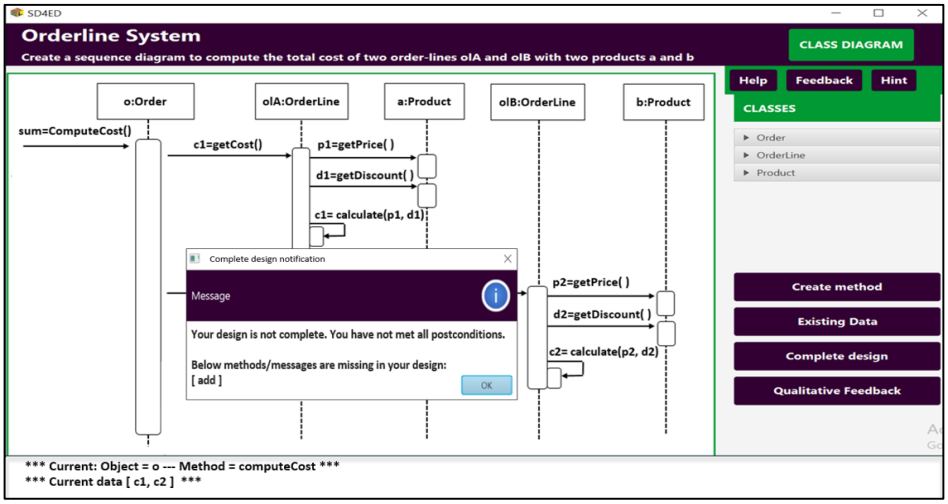

Figure 3. Example of feedback given if SD submitted without meeting all post-conditions specified in the use case.

Stage 3: Measuring the Quality of the Model

In the final stage qualitative feedback and marks are given based on quality metrics and students are allowed to improve their scores by resubmitting the design. Once students complete their design a holistic feedback on the quality of the design can be provided on completion of all the tasks based on the metrics. We do this by measuring the number of messages initiated by each interacting object (measuring the extent of distribution), giving the level of coupling/cohesion and any redundant messages used. We award higher marks when the number of messages dispatched by each entity are more evenly distributed, thus leading to low coupling and high cohesion. Moreover, our novel tool used a form of gamification to improve novices' motivation and engagement in modelling task. So, students will keep resubmitted their design in order to obtain a high mark for the quality of their design. Thus, this can helps novices to improve the learning outcomes in modelling concepts. An example of qualitative holistic feedback and scores provided to novices using our tool is shown in Figure 4, where a student had designed a centralized design with low cohesion.

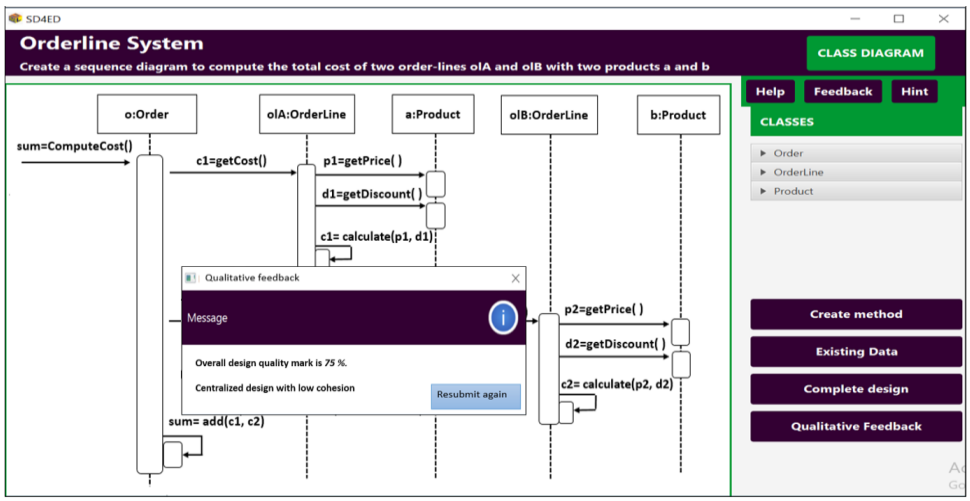

Figure 4. Example of qualitative feedback generated in the tool.

4. Designing of E-Learning Modelling Environment with CLT

The CLT assumes that learning and performance are impaired by overload on human working memory [27]. Modelling sequence diagram is one of the most cognitively demanding tasks among all the UML modelling as it involves a high number of interacting items that must be concurrently handled in the working memory. High cognitive loads for modelling sequence diagrams also stems from the need to consider the presence of links (association, local or parameter), availability of knowledge elements at the source object at current state, and possible paths to the goal state by constantly checking the interdependencies with class and use case diagrams. When qualitative measures are used, students also must consider how responsibilities can be delegated effectively by choosing appropriate pathways.

Our e-learning modelling tool for SDs followed CLT recommendations. One such recommendation we followed was to break down the complex task into meaningful parts or modules that can be processed and learned separately [12], [29]. Since the working memory treats each module as a single unit of information, our 'divide-and-conquer' approach has reduced the cognitive load. The problem space was subdivided into meaningful parts, reducing the number of interacting elements to be considered concurrently. Our framework permits the cognitive load to be managed by requiring students to focus only on consistency and constraints aspects before they consider post-conditions (completeness) in use case and qualitative aspects.

Poor usability of e-learning tools can negatively impact educational outcomes, especially when the material to be learnt is complex [26]. Poor usability also limits the potential advantage achieved from e-learning environments [31] by imposing an extraneous cognitive load, as learners struggle with the interface and challenges of the content presented. Some researchers have found significant learning effects from optimising the usability of learning materials [7]. This is most likely to be seen with novice modellers who experience the content as presenting a high intrinsic cognitive load and would therefore be more sensitive to any extraneous load imposed by poor usability [28]. Other researchers have reported improvements in efficiency, satisfaction or motivation [13], [31]. One of the main principles of good interface design suggested [17] is that to keep the learners informed through timely appropriate feedback and they also should always know where they are, which actions can be taken and how these actions can be performed. Thus, these factors have been addressed in our tool to help novices achieve better learning outcomes.

5. Results

This section reports on our results collected from the participants of our longitudinal studies carried out for more than four years on the use of our tool. Each year, we trialed our tool with different cohorts of students who enrolled in the same SEF course. In the second and fourth year, each student was asked to carry out a pre-test before interacting with the tool, followed by a post-test and a survey.

The pre- and post-tests revealed substantial improvement in learning outcomes and student satisfaction for all groups generally, and the improvement was mainly at the higher levels of the Bloom's taxonomy, in the application and analysis types of questions. Table 2 shows the impact by comparing the average marks for pre- and post-test results as well as the percentage gained in two groups. The first group was made up of undergraduate and postgraduate students in the second year of our trial, and the second group was made up of postgraduate students converting from other disciplines. In group 1, 68 of the 243 students enrolled in the course, volunteered to trial the tool and complete both tests and the survey were used in the analysis. Table 2 also shows the average pre-test mark was 46.25%, while the average for post-test was 61.25%, resulting in 32% improvement. In group 2, 32 of the 94 students enrolled in the course volunteered to trial the tool. The percentage improvements were higher at 57% as the average mark went up from 35 to 55.

Table 2. Summary of overall results from two different groups, noting the percentage gained between the pre- and post-tests in both groups used.

Data collected	Number of Students	Pre-Test Av.	Post-Test Av.	Pre-Test Std_Dev	Post-Test Std_Dev	Gain	Pre-Test Av. Completion Time (minutes)	Post-Test Av. Completion Time (minutes)
Group 1	68 (both)	46.25 %	61.25%	21.6%	25.8%	32%	13.26	10.25
Group 2	32 (postgraduate)	35%	55%	16.3%	22.6%	57%	14.07	12.18

To understand the tool's effectiveness on students for each SD concept, we analysed the results for each aspect of SD (consistency, completeness and qualitative questions). The pre- and post-tests also revealed improvement in understanding the interdependencies between class diagrams and SDs by the participating students. Table 3 shows the average marks for pre- and post-tests results for the questions related to interdependencies between class diagrams and SDs for two different groups. In group 1, the average score for the pre-test was 41%, while the average score for the post-tests was 51%, resulting in 26.3% improvement. The percentage improvements were higher at 96% and 55.56% for questions related to understanding the concepts of SD with use cases and qualitative aspects, respectively, as the average mark went up from 26% to 51% for the completeness questions and from 36% to 56% for the qualitative questions. Moreover, Table 3 also presents the results for group 2, where the average pre-test mark for the consistency questions was 35%, while the average post-test mark was 50.4%. The percentage improvements for this question in this group were 44%. For the completeness aspects questions, the average score for post-test increased to 37% as the pre-test was 19%, resulting in 94 % improvement. While for qualitative aspects questions, only 17.78% was the percentage improvements.

Table 3. Summary of results from both different groups for each aspect of SD (consistency, completeness and qualitative questions).

Data Collected	Number of Students	Pre-Test Av.			Post-Test Av.		
		Consistency Questions	Completeness Questions	Qualitative Questions	Consistency Questions	Completeness Questions	Qualitative Questions
Group 1	68	41%	26%	36%	51.8%	51%	56%
Group 2	32	35%	19%	45%	50.4%	37%	53%

We have analysed the students' individual log files in the tool, in order to identify how students learn the SD concepts during their interaction with the tool. Students were presented with multiple use cases in the tool. Table 4 illustrates the average number of error message provided when students violating the specified constrains as well as the completion time for the tasks in each attempt. The data shows a regular decrease of number of error message and time completion over the time as the comparison of the average marks for each attempts decreased.

Table 4. Summary of overall results from analysing tool log files for all students.

Average	First Attempt	Second Attempt	Third Attempt
Number of Error Messages	49.6%	26.6%	22.7%
Completion Time for Tasks	14.03 minutes	8.17 minutes	6.9 minutes

To understand the tool's effectiveness on students with different grades, we also analysed the distribution of student marks in pre- and post-tests for the group 1 (the large number of different cohorts of students who trailed the tool). We found that the number of students scoring in the range 0-49 declined by nearly 60% from 32 students to 13 students (as showing in figure 5), suggesting such modelling tools can significantly improve the performance of stragglers in the exam.

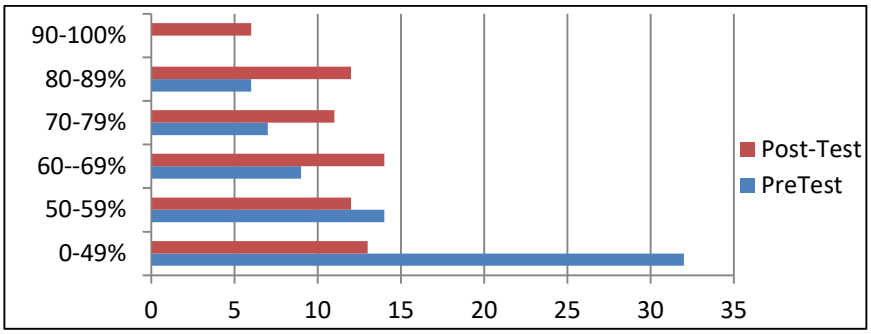

Figure 5. Marks Distribution in Pre/Post-tests for group 1.

Overall, participant performance in modeling SDs improved in tests especially for the straggler modelers, who were the main target group in our study. The improvement was specifically in understanding the relationship between modelling task use-cases, class diagrams and SDs. Students also found prompts and fading buttons increased interaction while allowing enough cognitive conflicts necessary for learning. Therefore, the CLT approach used in our tool allowed the cognitive load on novices to be reduced by requiring them to focus only on consistency constraints at each intermediate stage, before they ensure all use case post-conditions and qualitative aspects were met in their design.

5.1. Student Perception and Feedback

Student perceptions were measured using their feedback related to the tool through a specially designed survey. A questionnaire was designed to identify how the tool feedback could be improved in subsequent iterations and how they perceived the effectiveness of the tool in improving their design skills.

Table 5. The results of the questionnaire.

#	Questions	SD/D (%)	N (%)	SA/A (%)
1	The example scenarios presented were easy to follow.	9.09	27.27	63.6
2	I find sequence diagrams harder than most other UML diagrams.	9.09	9.09	81.82
3	The feedback given using the SD4ED Tool made me confident with sequence diagrams.	9.09	18.18	72.7
4	The Feedback feature on the overall design was useful.	0	27.27	72.7
5	The comments were easy to follow.	18.18	27.27	54.5
6	The incrementally complex design activities helped me to learn.	0	18.18	81.8
7	The system was intuitive and easy to use.	18.18	9.09	72.7
8	After using the tool I have a better understanding of sequence diagrams.	9.09	27.27	63.63
9	After using the tool I am better aware of interdependencies between class and sequence diagrams.	18.18	9.09	72.7
10	After using the tool I am better aware of how post-conditions in use cases are impact on sequence diagrams.	9.1	9.1	81.8
11	The system was responsive.	0	36.36	63.63
12	The Hint Features were useful.	27.27	27.27	45.4
13	Such a tool should be made available early in the semester.	0	27.27	72.7

In the analysis of the results we combine the “Strongly Disagree” and “Disagree” responses under SD/D category, and “Strongly Agree” and “Agree” responses into an SA/A category, respectively. Responses to the questions in table 5 revealed that the tool was particularly effective in helping novices identify and correct their misconceptions. Specifically, they found the immediate feedback based on the class diagram and uses case post-conditions supplied helped them to understand the interdependencies between class and sequence diagrams and impact of post-conditions for on SDs completeness. Over 72% of students had better understanding of interdependencies between classes and SDs after using the tool and receiving immediate feedback on their SD design. Over 81 % of students had better awarding of how post-conditions in use cases are impact on sequence diagrams.

Students were also surveyed with open-ended question. Overall, the responses showed students had a better understanding of sequence diagrams, as showing in table 7. Responses to the question asked in table 6 show that most students found the feedback mechanism provided at each stage helped them to grasp the design skills and correct their invalid mental models.

Table 6. Written feedback from the open-ended survey question.

Q. What aspects of the tool SD4ED did you like?	"It was pretty intuitive for the most part. The different levels of complexity were useful."
	"The current object and methods and what current data are exist in each stage."
	"The selection list, the prompt of the current object and method."
	"The quick and immediate feedback we got for several aspects."
	It was very responsive, it showed visually what "call a method" looks like."
	"I can practice in my own time and get feedback."
	"We can practice with examples and mistakes can be immediately corrected by the tool."
	"I really like the feature where I can check my sequence diagram is complete or not."

6. Discussion

Modelling sequence diagram were shown to be one of the most cognitively demanding tasks among all the UML modelling suite capturing different perspectives. High cognitive loads for modelling sequence diagrams stems from the need to consider the presence of links (association, local or parameter), availability of knowledge elements at the source object at current state, and possible paths to the goal state by constantly checking the consistency with class and completeness with use case post-conditions. When qualitative measures are used, students also must consider how responsibilities can be delegated effectively by choosing appropriate pathways. Our model has provided a way to structure the cognitive loads to suit diverse students by requiring students to focus only on consistency and constraints aspects through instant feedback at any stage by aggregating all past interactions before they consider post-conditions (completeness) and qualitative aspects. These decomposition features together with cohort-specific prompting, diagnostic feedback messages and ability to move to the goal states in steps provide a level of support closer to human tutors.

The lack of resources for teaching and learning as well as ever increasing class sizes have substantially reduced the opportunities for formative feedback, resulting in a high number of invalid and poor-quality student designs. These problems have been made worse by increasing reliance on online learning in recent years and the current lockdowns introduced in response to COVID'19. Lack of resources and limited face-to-face interaction, therefore, have made development of pedagogical tools inevitable. These problems are exacerbated for software engineering courses where increasing number of students come from many different disciplines. Most such students, especially those lacking experience with object-oriented programming find UML modelling and different UML diagrams very abstract. In the initial stage, our pedagogical tools therefore aimed to reduce the high cognitive load in modelling SDs by giving feedback on consistency with class diagrams and then whether responsibilities in use-case diagram are correctly discharged. Our knowledge-based approach guiding students to create consistent and complete UML diagrams have helped stragglers overcome their initial mental blocks.

Though developing e-learning tool require CLT and a longitudinal study, our pre- and post-tests with different- group of students reveal such an approach can help improve the learning outcomes, especially for the strugglers. Our feedback shows high cognitive load needed having to consider various aspects of sequence diagrams concurrently had been one of the main reasons for past student difficulties. Sequence diagrams play a key role with modern software development which requires modelling the interaction between objects using a number of different frameworks as with full stack development. Moreover, the design patterns promoting qualitative attributes such as reusability, extensibility, maintainability are expressed mainly through sequence diagrams. Hence, we believe more research is needed to develop pedagogical that can lower the reliance on human tutors for novices. The context of our research, however, is not to replace the existing modelling tools, but to supplement them. Our research is focused on finding ways of helping novice designers to get over the initial learning barriers before proceeding to use tools improving productivity, in subsequent courses.

7. Limitations of the Study

Our e-learning tool incorporating CLT has shown substantial improvement to SD design related learning outcomes, however, there are a number of other factors that may have influenced the results. The number of students volunteering to take part in these activities remain relatively small as the data collection period usually coincide with students' peak periods. The self-selecting students in any study may not be reflective of the actual cohort, but the grade distribution of participants and non-participants was broadly similar in terms of average and standard deviation, in the mid-semester test prior to the study. The poor usability of the versions used in the first group trailed may have caused some students not to proceed with the post-test and the survey, but usability of the agent for version 2 has greatly improved. Though students perform better when cognitive load is reduced by problem decomposition and scaffolding, it is not clear to what extent students will be able to solve complex modelling problems when no support is provided.

8. Conclusions

Modelling sequence diagrams poses heavy cognitive load on students as constraints and rules imposed by other UML models must be analyzed concurrently, making it the most poorly understood UML artefact. Our approach better manages cognitive load by subdividing a complex problem into independent meaningful parts with reduced number of interacting elements. In the initial stages, students have to focus only on one or two aspects. Our e-learning modelling tool automatically enforces consistency checking, checks for completeness and gives qualitative feedback based on domain models, use cases and quality metrics specified by the instructor. The theoretical design and pedagogy for the proposed tool is based on CLT.

The extensive automated feedback enables large diverse groups of students to learn to model sequence diagrams. The e-learning tool manages cognitive load by varying the type and amount of help as a student progresses in modelling tasks. In the first stage, invalid and inconsistent messages are prevented from being dispatched while giving immediate feedback. In the next stage, submission is disabled until the design itself is valid, and if invalid, feedback is given on constraints violated or unmet. In the final stage, qualitative feedback on submitted design is provided based on specified metrics. The longitudinal study allowed data collected from experienced tutors, lecturers and participants to evolve a more personalized teaching approach better suited to our increasingly diverse student cohorts. The e-learning modelling tool developed has shown substantial improvements in learning outcomes and student satisfaction. One main benefit of our approach is that it can be easily replicated for other complex modelling tasks. In our future work, we plan to trial our tool with different cohorts of students and other UML models.

References

1. Agner, L.T., T.C. Lethbridge, and I.W. Soares.: Student experience with software modeling tools. Software & Systems Modeling. 18(5): p. 3025-3047 (2019)
2. Agner, L.T. and T.C. Lethbridge.: A survey of tool use in modeling education. In 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS). IEEE (2017)
3. Alhazmi, S., C. Thevathayan, and M. Hamilton.: Learning UML Sequence Diagrams with a New Constructivist Pedagogical Tool: SD4ED. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (2021)
4. Alhazmi, S., C. Thevathayan, and M. Hamilton.: Interactive Pedagogical Agents for Learning Sequence Diagrams. In International Conference on Artificial Intelligence in Education. Springer (2020)
5. Alphonce, C. and P.: Ventura. QuickUML: a tool to support iterative design and code development. In Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (2003)
6. Auer, M., T. Tschurtschenthaler, and S. Biffl.: A flyweight uml modelling tool for software development in heterogeneous environments: IEEE (2003)
7. Avouris, N.M., Dimitracopoulou, A., Daskalaki, S. and Tselios, N.K.: Evaluation of distance-learning environments: Impact of usability on student performance. International Journal of

Educational Telecommunications. **7**(4): p. 355-378 (2001)
8. Bannert, M.: Managing cognitive load—recent trends in cognitive load theory. Learning and instruction. **12**(1): p. 139-146 (2002)
9. Bloom, B.S.: Taxonomy of educational objectives. Vol. 1: Cognitive domain. New York: McKay. **20**: p. 24 (1956)
10. Dranidis, D., I. Stamatopoulou, and M. Ntika.: Learning and Practicing Systems Analysis and Design with StudentUML. In Proceedings of the 7th Balkan Conference on Informatics Conference (2015)
11. Eckerdal, A., McCartney, R., Moström, J.E., Ratcliffe, M. and Zander, C.: Can graduating students design software systems? ACM SIGCSE Bulletin. **38**(1): p. 403-407 (2006)
12. Gerjets, P., K. Scheiter, and R. Catrambone.: Designing instructional examples to reduce intrinsic cognitive load: Molar versus modular presentation of solution procedures. Instructional Science. **32**(1): p. 33-58 (2004)
13. Hancock, P.A., A.A. Pepe, and L.L. Murphy.: Hedonomics: The power of positive and pleasurable ergonomics. Ergonomics in design. **13**(1): p. 8-14 (2005)
14. Kirschner, P.A.: Cognitive load theory: Implications of cognitive load theory on the design of learning, Elsevier (2002)
15. LeBlanc, R.J., Sobel, A., Diaz-Herrera, J.L. and Hilburn, T.B.: Software engineering 2004: curriculum guidelines for undergraduate degree programs in software engineering.: IEEE Computer Society (2006)
16. Lethbridge, T.C.: Teaching modeling using Umple: Principles for the development of an effective tool. In 2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T). IEEE (2014)
17. Nielsen, J.: Ten usability heuristics, http://www. nngroup. com/articles/ten-usability-heuristics Accessed June 17, 2021
18. Paas, F., A. Renkl, and J. Sweller.: Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. Instructional science,. 32(1/2): p. 1-8 (2004)
19. Pourali, P. and J.M. Atlee.: An experimental investigation on understanding the difficulties and challenges of software modellers when using modelling tools, Technical Report CS-2018-03. David R. Cheriton School of Computer Science (2018)
20. Pourali, P. and J.M. Atlee.: An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools. In Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. (2018)
21. Rayner, M., Hockey, B.A., Chatzichrisafis, N. and Farrell, K.: OMG unified modeling language specification. In Version 1.3,© 1999 Object Management Group, Inc. Citeseer (2005)
22. Schaffer, H.E., Young, K.R., Ligon, E.W. and Chapman, D.D., Automating individualized formative feedback in large classes based on a directed concept graph. Frontiers in psychology. **8**: p. 260 (2017)
23. Sien, V.Y.: An investigation of difficulties experienced by students developing unified modelling language (UML) class and sequence diagrams. Computer Science Education. **21**(4): p. 317-342 (2011)
24. Sin, T.: Improving novice analyst performance in modeling the sequence diagram in systems analysis: A cognitive complexity approach, Florida International University (2009)
25. Song, I.-Y., Khare, R., An, Y., and Hilsbos, M.: A multi-level methodology for developing UML sequence diagrams. in International Conference on Conceptual Modeling. Springer (2008)
26. Sweller, J.: Cognitive load theory, learning difficulty, and instructional design. Learning and instruction. **4**(4): p. 295-312 (1994)
27. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive science. **12**(2): p. 257-285 (1988)
28. Sweller, J., J.J. Van Merrienboer, and F.G. Paas.: Cognitive architecture and instructional design. Educational psychology review,. **10**(3): p. 251-296 (1998)
29. Syn, T. and D. Batra.: Improving Sequence Diagram Modeling Performance: A Technique Based on Chunking, Ordering, and Patterning. Journal of Database Management (JDM). **24**(4): p. 1-25 (2013)
30. Van Merrienboer, J.J. and J. Sweller.: Cognitive load theory and complex learning: Recent developments and future directions. Educational psychology review. **17**(2): p. 147-177 (2005)
31. Zaharias, P.: Usability in the context of e-learning: A framework augmenting 'traditional' usability constructs with instructional design and motivation to learn. International Journal of Technology and Human Interaction (IJTHI). **5**(4): p. 37-59 (2009)