# Common Phenomena Exhibited by Students in Their Individual Design Processes: a Multi-Scenario Case Study on Software Design Education

T. Fu, R. Sun, C. Li, and L. Wang

*Abstract*—**Cultivating students' software design capabilities through effective training has always been a challenge in software education. This paper is aimed at addressing this issue by adopting a multi-scenario case study approach to examine the independent design processes of 23 undergraduate students on an online teaching system. The selected case scenario models include transaction flow diagrams (TFDs), activity diagrams, sequence diagrams, and entity–relationship (ER) diagrams. By analyzing students' behavioral performances and design outcomes, a series of common phenomena are identified. These phenomena encompass common errors, such as overlooking key steps, struggling to distinguish similar data objects, and omitting critical entities or attributes. Common behaviors include offering various solutions, facing challenges in achieving specific design goals due to a lack of prior experience, and experiencing difficulties meeting requirements using prescribed syntax. Common approaches to assist students include providing reference software, adopting teamwork for idea generation, and allowing iterative modifications to improve outcomes.**

**Based on these common phenomena exhibited by students, several recommendations are provided for software educators to enhance the development of students' software design capabilities, which mainly include considering students' prior experience in assignments, providing design references for unfamiliar software, encouraging peer discussions and multiple iterations, and guiding students towards continuous improvement rather than disregarding unconventional outcomes.**

**The common phenomena identified in this paper seamlessly integrate with software design education, reflecting its distinct characteristics. Those common phenomena will help researchers understand student needs and challenges. Additionally, the research design, which analyzes student behaviors based on their software design outcomes, provides a fresh perspective in the field. Furthermore, the conclusions drawn in this paper offer valuable insights for educators who aim to enhance classroom experiences in software design courses.**

*Index Terms*—**Object-Oriented Systems Analysis and Design, Software Design Education, UML**

## I. Introduction

Software design capability is crucial for the future career development of students majoring in information management and information systems and software engineering, as those students are expected to provide robust and extensible solutions to various real-world software requirements [1]. Many undergraduate courses, such as information systems analysis and design and software engineering, are offered to train and enhance students' software design capability. However, we must admit that students often face challenges in those software design courses [2][3][4]. Correspondingly, software design education has become a controversial research area in academia, with numerous articles on teaching techniques and methods [5][6], students' cognition and attitudes [7][8], and supporting tools [9][10][11]. Notably, there is a lack of research on the common phenomena exhibited by students when they independently complete their software design processes and the underlying causes of these occurrences.

Software design courses such as information systems analysis and design and software engineering primarily teach students how to generate and describe various solutions in the process of developing information systems or software using unified and standardized paradigms, rather than providing specific computer program code to implement these solutions. Undoubtedly, software design is the necessary precursor to programming [12]. Although the programming directly produces software or information system products, the quality of these products is primarily

This work was supported in part by Beijing Natural Science Foundation under Grant 9222010, and by Humanities and Social Science Foundation of Ministry of Education of the People ’ s Republic of China under Grant 20YJCZH066.

T. Fu and L. Wang are with Economics And Management School, Beijing University of Technology, Beijing 100124, China. (e-mail: futao@bjut.edu.cn) (e-mail: wanglian@bjut.edu.cn).

R. Sun is with Computer School, North China Institute of Aerospace Engineering, Langfang 065000, China. (e-mail: sunran@nciae.edu.cn).

C. Li is with Economics and Management School, North China University of Technology, Beijing 100144, China. He is the corresponding author. (e-mail: lichenguang@ncut.edu.cn).

determined by the software design [13][14]. Courses on software design are typically scheduled after the completion of courses on programming and database management [15], which hints that the curriculum of software design courses is based on the content taught in the programming and database management courses. Moreover, software design courses also impose certain requirements on students' comprehensive application of programming and database management knowledge.

Structured design and object-oriented design are commonly utilized software design methodologies. However, object-oriented design offers a multitude of advantages, including heightened productivity, reduced maintenance concerns, expedited software development, improved adaptability to changes, and enhanced system quality [16][17]. Consequently, object-oriented design has emerged as the predominant approach in software design and is frequently incorporated into software design curriculum. Unified Modeling Language (UML), which provides the most mainstream paradigm for object-oriented software design, was developed by Grady Booch, James Rumbaugh, and Ivar Jacobson in the mid-1990s, and became the Object Management Group (OMG) standard and the de facto industry standard [3][18]. UML provides many types of diagrams to model different aspects of the target information system or software. UML diagrams alone are still insufficient to cover the entire software design process [19], and additional diagrams, such as transaction flow diagrams (TFDs) and data flow diagrams (DFDs), are still needed as supplements. Of course, those different aspects of the target information system or software as well as their related diagrams are interconnected. Analyzing a group of students' individual design processes for the same target information system or software and comparing their independent design results (namely, UML and other diagrams) regarding several key aspects to identify common phenomena, will enable us to understand students' real thoughts and common practices in their software design processes and help us grasp key points for improving students' software design capabilities. All of these, in turn, will enhance the teaching effectiveness of those software design courses.

The remainder of this paper is structured as follows: in Section II, we provide an overview of related research that is primarily focused on student learning behaviors and the associated methodologies. This literature review serves as a theoretical foundation for the methodology employed in this paper. In Section III, we present the research design in detail, including information about the specific course under investigation, the participants, and the step-by-step procedure followed throughout this multi-scenario case study. In Section IV, we delve into the selection of multiple software design models, the creation of corresponding case scenarios based on students' actions during model construction, and the analyses of these scenarios. Our aim here is to identify recurring phenomena in students' behaviors. Then in Section V, we summarize and analyze the observed common phenomena and collect student feedback through a questionnaire specifically targeted at these phenomena. In Section VI, we discuss the limitations and challenges of this multi-scenario case study and suggest further research directions. Ultimately, we draw conclusions about the observed common phenomena and provide recommendations for software educators.

## II. LITERATURE ON STUDENT LEARNING BEHAVIORS

Although research specifically addressing common behavioral phenomena among students in software design education is limited, a substantial body of literature in the field of education explores student learning behaviors [20][21]. Discussions regarding the impact of student learning behaviors on student academic performance can be traced to earlier decades (e.g., [22][23]) and comprise a prominent research area. In the broader educational context, student learning behaviors typically encompass various activities related to the learning process, including student communication, interaction, self-motivation, and satisfaction with education [24][25]. These learning behaviors can be observed in any educational setting but may not necessarily encompass the distinctive characteristics of a specific educational domain.

Regarding research methods and tools, many studies on student learning behaviors rely on questionnaires or self-reports to directly investigate whether students have engaged in specific types of behaviors during their learning process. These inquiries also extend to students' perceptions and feelings regarding these learning behaviors, as well as the impact of such behaviors on their academic performance (e.g., [26][27]). However, the limitation of such research is the challenge of eliminating subjectivity introduced by the students themselves [28]. Alternatively, some studies employ trace logs and the case study method to record the specific behaviors exhibited by students during the learning process (e.g., [29][30]). While these approaches effectively mitigate subjectivity, they often cannot predict which learning behaviors will occur prior to tracking and recording. Additionally, such studies typically require a substantial number of subjects and an extended tracking period to observe valuable learning behaviors [31][32].

Concerning the specific environments in which student learning behaviors take place, the majority of the literature primarily concentrates on student learning behaviors within traditional classroom settings (e.g., [26][33][34]). However, in recent years, there has been a growing body of literature dedicated to discussing student learning behaviors in online learning environments, particularly within the context of massive open online courses (MOOCs) (e.g., [31][35]), facilitated by the robust data collection capabilities of internet-based educational platforms. Furthermore, some studies endeavor to analyze student learning behaviors in blended learning environments, which seamlessly integrate offline classroom teaching with online learning components (e.g., [21][36]).

In summary, existing research on student learning behavior has provided valuable insights and tools. However, the common phenomena exhibited by students in software design vary significantly from the generalized student learning behaviors defined in the field of educational research. These phenomena need to be closely integrated into the software design education process and accurately reflect the unique characteristics of software design education. Currently, there is limited research specifically examining the common phenomena exhibited by students in their software design processes, which provides opportunities for this paper.

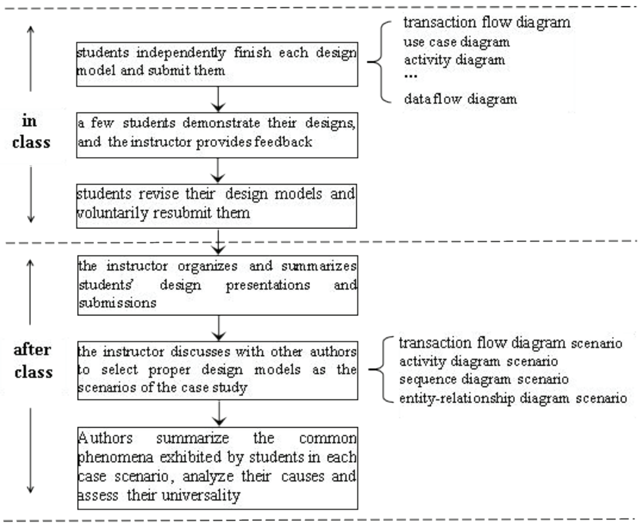
## III. METHODOLOGY

The primary research method employed in this paper is a multi-scenario case study. The case focuses on the Object-Oriented Information Systems Analysis and Design course, which primarily teaches object-oriented information system design methods and models. The main objective of this course is to foster students' capabilities in designing information systems or software, and it is one of the core courses in undergraduate information management and information systems programs. At Chinese universities, information management and information systems majors commonly spend 48 academic hours to learn this course. The first author of this paper has been continuously teaching this course for nearly 15 years. In each cycle, after teaching models on requirements analysis, system analysis, system design, system implementation, and system testing, he typically allocates 10 academic hours for all students in the class to independently design the same target information system. The case study described in this paper primarily draws upon the students' independent design stage from the last of the teaching cycles (just completed in 2023).

The course was given to 23 undergraduate students who selected information management and information systems as their minor program. Their target system was a simple online teaching system. The course instructor (also the first author of this paper) provided students with only a prompt that the system consists of four subsystems: user management subsystem, online teaching subsystem, assignment management subsystem, and exam subsystem. The details of each subsystem were designed and developed by the students themselves. Students were required to independently conduct a requirements investigation, system analysis, and system design to provide corresponding related models, including TFDs, use case diagrams, activity diagrams, state diagrams, sequence diagrams, overall class diagrams, three formalizations of entity class attribute tables, ER diagrams, data dictionaries, and data flow diagrams.

The students' design status of each specific model could serve as an alternative scenario in this case study. The related teaching process in the classroom included the following steps: First, all students independently completed the model design and submitted the results via email. Afterwards, the instructor randomly selected one student to present its design to the entire class. The instructor then provided feedback, pointing out errors and deficiencies in the design. In addition, the instructor allowed 1 to 2 other students to voluntarily present their designs, which might be significantly different in style from the previous design, and provided feedback. After listening to the instructor's reviews, the students could choose to revise their own designs and resubmit them to the instructor via email (or choose not to make revisions).

After class, the instructor organized and summarized the design presentations and other design submissions from students. He then discussed the overall results with the other three authors of this paper. All authors are university teachers; the first, second, and fourth authors majored in information management and information systems as undergraduate students, and the third author specialized in software engineering for their master's degree. Additionally, all authors have taught multiple rounds of software design courses in fields such as information systems analysis and design, software engineering, or UML modeling. All authors together chose the case scenario models based on their discussions. The common phenomena exhibited by students in each case scenario were further assessed by all authors based on previous teaching experiences to determine their universality. The procedure of the multi-scenario case study in this paper is illustrated in Fig. 1.



in class
students independently finish each design model and submit them
transaction flow diagram
use case diagram
activity diagram
...
dataflow diagram
a few students demonstrate their designs, and the instructor provides feedback
students revise their design models and voluntarily resubmit them
after class
the instructor organizes and summarizes students' design presentations and submissions
the instructor discusses with other authors to select proper design models as the scenarios of the case study
transaction flow diagram scenario
activity diagram scenario
sequence diagram scenario
entity-relationship diagram scenario
Authors summarize the common phenomena exhibited by students in each case scenario, analyze their causes and assess their universality

**Fig. 1.** Multi-scenario case study procedure

## IV. MULTI-SCENARIO CASE STUDY

According to students' behavioral performances and design results for each model in the classroom and the subsequent discussions among the authorship, TFDs, activity diagrams, sequence diagrams, and ER diagrams have been selected as the scenarios for the case study in this paper. The common phenomena exhibited by students in these scenarios are acknowledged by all authors to have a certain degree of universality.

### *A. TFD scenario*

A TFD utilizes predefined graphical symbols and connectors to represent specific business flows [37][38] and illustrates the relationships between personnel and transactions, the sequence of transactions, and the flow of management information. A TFD is not a UML model. Because UML lacks a comprehensive model for describing specific transaction flows of enterprises or other organizations, some Chinese textbooks (e.g. [39][40]) consider TFDs as the first model for information systems or software requirements analysis. TFDs should be transformed into use case diagrams according to specific rules, which depict a static structure of an information system or software.

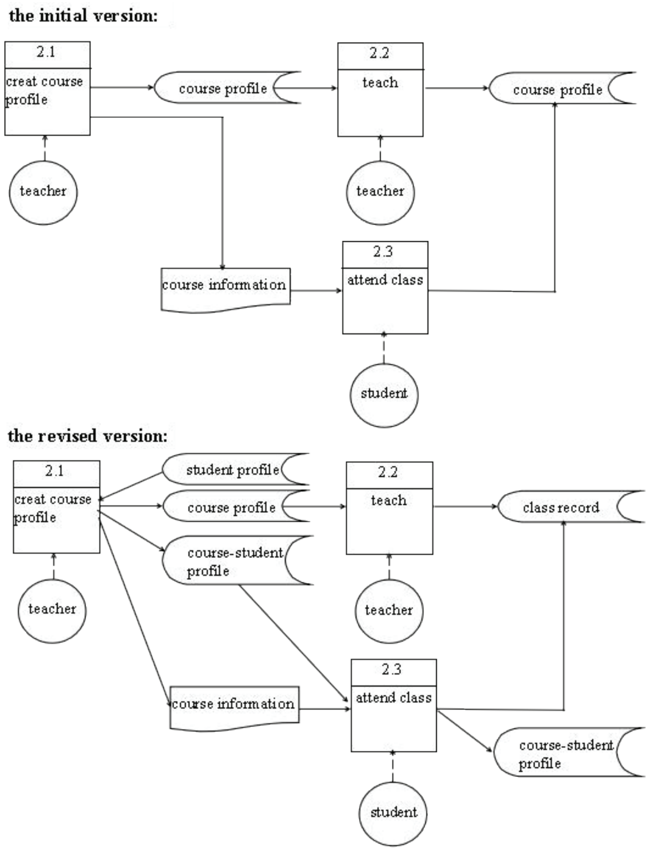


**Fig. 2.** TFDs of online teaching subsystem designed by a student giving the presentation

Fig. 2 presents TFDs of the online teaching subsystem designed by one of the students who gave a presentation. Her initial version demonstrates a clear and concise functional flow. Initially, the teacher establishes a profile for the course and sends relevant class information to the students. Then, the teacher conducts online classes and generates class records while the students attend classes as required, recording attendance information such as the number of classes attended and the duration of each class. The flaw in the original design, and a prevalent issue among novice software designers, is their inability to precisely distinguish similar data storage objects. Actually, this subsystem encompasses three types of data storage objects related to the course: course information (including course ID, course name, course type, academic year, weekly class schedule, teaching instructor, total hours, and other attributes), class record (including start time, duration, attendance count, and other attributes for each class), and student performance in a particular course (including accumulated attendance count, accumulated class time, and other attributes). The student only used the course profile to describe all three types of data storage objects, indicating a lack of distinction among these three data objects in her mind. After receiving feedback from the instructor, the student promptly understood the differences between them and made the necessary corrections (please refer to the revised version in Fig. 2).

Table I presents the statistics of student design errors for each model and reveals that none of the students' initial TFDs were perfect. Further analysis reveals that all students in the class had difficulty distinguishing similar data storage objects. After listening to the feedback provided by the instructor, 20 students chose to resubmit their modified designs. Among them, 10 students demonstrated an improved understanding of properly defining data storage objects, resulting in nearly perfect designs. Note that six students decide to generate a class-student record in their designs, indicating their belief in recording the performance of each student during every class session. This differs from the showcased design, which only utilized the course-student profile to track cumulative attendance and learning duration. This variation highlights the diversity of solutions that students develop when they are confronted with the same problem.

TABLE I
STATISTICS OF STUDENT DESIGN ERRORS

| model | first submission | | | second submission | | |
|---|---|---|---|---|---|---|
| | nearly perfect | few errors | many errors | nearly perfect | few errors | many errors |
| TFD | 0 | 21.7% | 78.3% | 50.0% | 50.0% | 0 |
| activity diagram | 8.7% | 43.5% | 47.8% | 84.2% | 15.8% | 0 |
| sequence diagram (team design) | 52.2% | 47.8% | 0 | — | — | — |
| ER diagram | 0 | 34.8% | 65.2% | 65.2% | 26.1% | 8.7% |

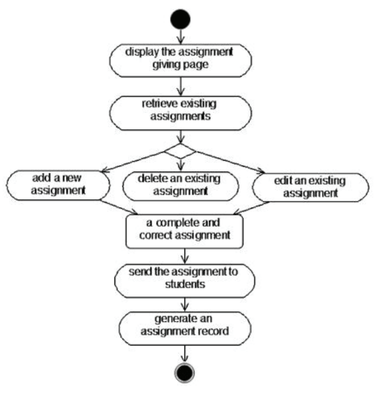
### *B. Activity diagram scenario*

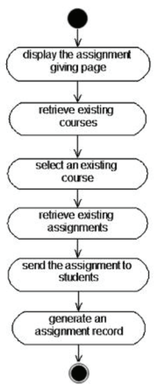
An activity diagram is one of the UML models that can be utilized to describe the sequence of internal activities within a use case [41][42][43]. In this scenario, the instructor selected a leaf use case from each subsystem of the target system and asked all students to draw activity diagrams for the four selected leaf use cases. Two students were chosen to present their designs for the "Assignment Management" subsystem's "Create Assignment" use case, as shown in the upper part of Fig. 3. The initial versions of their designs highlight common errors. The initial version (a) demonstrates all the activities performed by the teacher, from opening the assignment giving page to distributing a complete and correct assignment to students. However, this version overlooks the fact that the instructor may be responsible for multiple courses and should select a course before performing assignment operations. The initial version (b) addresses this issue by selecting a course before retrieving existing assignments. Unfortunately, this design fails to include the addition or modification of an assignment before sending it to students, which does not align with reality. After reviewing each other's designs, the two students collaboratively drew the revised version shown in the lower part of Fig. 3, which represents a complete and correct solution. This scenario illustrates that when the object function (represented by the use case) involves multiple steps or activities, students are prone to overlooking certain activities or steps. However, if students reference each other's designs, they have the opportunity to reduce such omissions.

Based on the design results of all 23 students in the class (please see Table I), it is evident that two students managed to avoid the mentioned omissions in their initial designs and achieved near perfection, while the majority of students faced difficulties in this aspect.

Initial versions:

(a) (b)


display the assignment giving page
retrieve existing assignments
add a new assignment
delete an existing assignment
edit an existing assignment
a complete and correct assignment
send the assignment to students
generate an assignment record


display the assignment giving page
retrieve existing courses
select an existing course
retrieve existing assignments
send the assignment to students
generate an assignment record

Revised version:

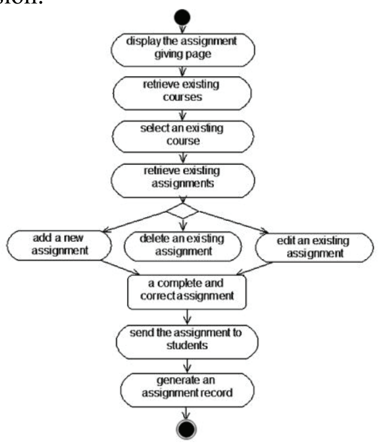
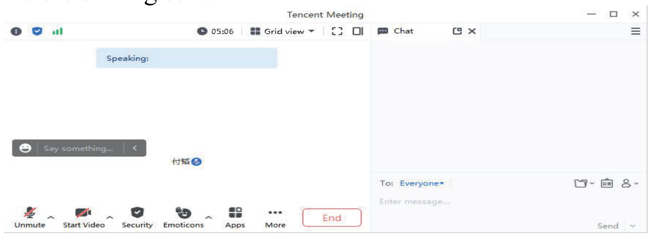

display the assignment giving page
retrieve existing courses
select an existing course
retrieve existing assignments
add a new assignment
delete an existing assignment
edit an existing assignment
a complete and correct assignment
send the assignment to students
generate an assignment record

**Fig. 3.** Activity diagrams of the use case "give assignment" designed by students giving presentations.

### *C. Sequence diagram scenario*

The sequence diagram is the most essential UML model in the system analysis phase. This model primarily describes the various objects involved in the leaf use case and the communication between them [44][45][46]. These objects will be transformed into concrete classes in subsequent class analysis models. The basic building block of object-oriented program code is classes, and the identification of classes in the target information system mainly relies on sequence diagrams and class analysis models. In this scenario, the instructor initially requested that all students should independently draw a sequence diagram for the use case "teach" in the online teaching subsystem within one class session. However, after some contemplation, the students responded that although they frequently attended online classes, they had not utilized the teacher's online teaching functions. Therefore, the instructor suggested that the students refer to the software "Tencent Meeting", as it has interfaces and functions that are identical for both teachers and students (see Fig.4 for its interfaces). By using these functions, online teaching can be adequately supported. The students then expressed that they had a rough design idea by referring to Tencent Meeting. However, since Tencent Meeting is a standard industry software with relatively complex functions, the students actively requested the instructor's permission to complete the sequence diagram as a team and extend the design time. Eventually, the 23 students spontaneously formed six teams, with each team consisting of three to five students. All teams utilized one week of design time after the class to complete their designs and presented them in the next class session.

Before sharing screen



After sharing screen

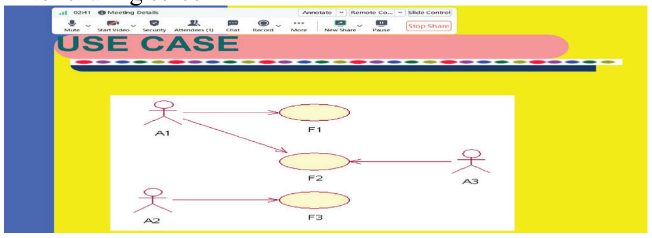


**Fig. 4.** Operating interfaces of Tencent Meeting.

According to the team work presentations, it is evident that completing the design as a team and having more ample design time indeed improved the students' design effectiveness. Among the six teams, three teams achieved near perfection with minimal errors, while the remaining three teams had only a few errors (see Table I). Fig. 5 presents the well-executed design of one team, which includes three boundary objects and three entity objects, achieving functions including teacher's class start and end time recording, sharing of teacher's screen, microphone control, and public screen chatting during class. Although the sequence diagram of this design is quite complex, it does not include functions such as transmitting teacher's video images and private messaging. The class and chat records in this design are generated based on specific course files, indicating that each class record and chat message must be associated with a particular course. Other students' designs associate chat records directly with the class, implying that public screen chatting can only occur during class time, which is also a reasonable design. All teams emphasized that the design process is an incremental improvement, and the team's design results underwent multiple iterations and modifications to achieve the final version. Any team member alone could not achieve such results within a single class session. We consider the design depicted by Fig. 5 as an example. Initially, the design only depicted the action sequence of sending messages (see operations 19 to 23 in Fig. 5). However, team members had a lingering feeling that something was missing. After careful consideration, a student eventually pointed out that it only depicted sending messages but did not include receiving messages. This statement was immediately agreed upon by other team members, and the function of receiving messages was added to the diagram (see operations 14 to 18 in Fig. 5).

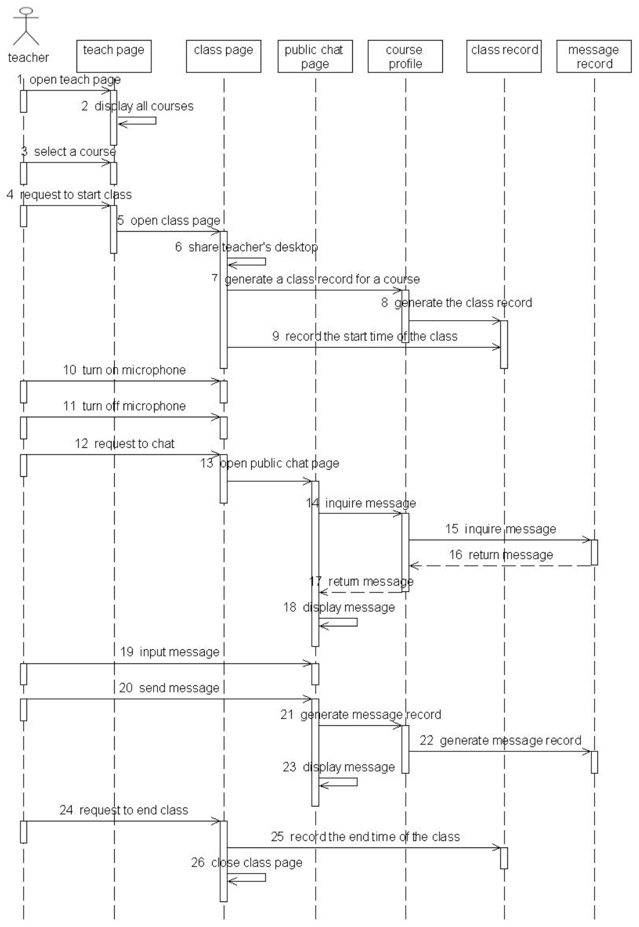


**Fig. 5.** Sequence diagram of the use case "teaching" designed by a student team

Although teams' designs were generally superior to individual designs, given that those team designs cannot reflect each individual student's design level and that some students tend to "free-ride", this course only allowed students to adopt team design for very complicated models such as sequence diagrams.

### *D. ER diagram scenario*

An ER diagram depicts the attributes of entity classes and the relationships between them [47][48]. This is a central model in the field of information system or software data design. There are various formats for ER diagrams [47], and this course primarily focuses on IDEF1X diagrams [49][50]. Considering the difficulty of the design, the instructor required students to independently draw an IDEF1X diagram that excludes entities related to the public screen chatting function. Students were also allowed to omit less significant attributes, with the exception of primary keys and foreign keys.

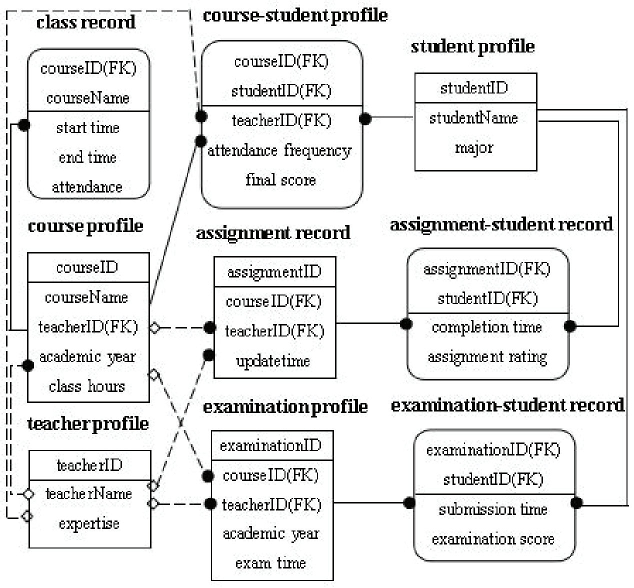


**Fig. 6.** Modified version of the IDEF1X diagram designed by a student giving the presentation. This diagram omits public screen chatting-related entities and many insignificant attributes of other entities.

The statistical results in Table I indicate that no student was able to provide a near-perfect design and that students made a wide range of errors. Here, we can only show a revised version of the ER diagram in Fig. 6 by a student after his initial design was reviewed by the instructor. For types of errors encountered, the most common mistake was omitting key entities or certain crucial attributes of entities. For example, some students failed to create an "examination–student record" entity, resulting in the inability to record the submission time and exam scores of students. Some students also omitted the "final score" attribute in the course-student profile entity, which prevented the fulfillment of the requirement that the course score be determined by a combination of the attendance frequency, assignments, and exam score. In essence, these errors reflect students' incomplete solutions that do not meet the system requirements. Nearly all students made these mistakes in this scenario. Additionally, there were other common errors that indicated a lack of understanding of diagramming rules, such as incorrect usage of ER symbols and forgetting to label foreign keys. However, diligent students who seriously approached the task tended to avoid such errors. In this case, nine students did not make these types of mistakes. Overall, this scenario demonstrates the challenge in information systems or software design, which requires students to provide comprehensive solutions that meet system requirements while expressing them correctly based on syntax and rules. It is generally more difficult for beginners to present comprehensive solutions than to avoid grammar and rule-related errors.

TABLE II
COMMON PHENOMENA AND THEIR EXPLANATIONS

| No. | common phenomena | types | explanations |
|---|---|---|---|
| 1 | Students often overlook key steps or activities when the targeted function involves numerous steps or activities. | common errors | In the field of software design, it is common to have target functions that involve many steps or activities. Students are required to separately identify each of these steps or activities in the order in which they occur, without missing any. This process can be quite challenging for beginners. |
| 2 | Students often struggle to distinguish similar data storage objects in data design. | common errors | Many entities inherently share similarities, such as a course, a course offered in a specific academic year, and a class within that course, all of which can be considered entities. Students can easily become confused by these entities. |
| 3 | Students may disregard critical entities or omit essential entity attributes. | common errors | Entities and entity attributes are typically linked to specific functionalities, and an information system typically encompasses numerous functions. If students overlook certain functions when designing entities, they can easily miss the entities or entity attributes associated with them. |
| 4 | Students may provide a variety of solutions when faced with the same design requirement. | common behaviors | The specific solutions students offer for a given design requirement depend on their prior knowledge and what aspects of the problem they concentrate on. Variations in prior knowledge and focus can lead to diverse solutions, but these solutions may all fulfill that requirement. |
| 5 | Student may face difficulties in providing comprehensive solutions when tasked with designing related functions and data, if they have not previously employed a certain type of information system or software. | common behaviors | Lacking experience with a specific type of information system or software means students must independently envision the structure of the target system or software and all its functional details, which heightens the design complexity. |
| 6 | The challenge in software design lies in meeting requirements while accurately expressing them according to prescribed model syntax and rules, and for novices, achieving the former is more difficult. | common behaviors | For students, generating solutions that meet requirements usually requires practical software development experience, while accurately expressing these solutions often just necessitates attentive classroom participation. Students' primary deficiency is hands-on software development experience. |
| 7 | Providing students with ready-to-use software for reference can simplify their design process. | common supportive approaches | In this way, students can imitate various design aspects of existing software, which reduces their mental load. However, directly copying design elements from existing software also limits opportunities to foster their creativity. |
| 8 | Teamwork can facilitate the generation of more design ideas and help minimize omissions. | common supportive approaches | Individuals may face various design challenges. In a collaborative team setting, those proficient in a specific area can aid teammates who encounter difficulties, and team members can also mutually review and rectify errors. |
| 9 | Allowing students to engage in multiple iterations and repeated modifications can significantly improve their design outcomes. | common supportive approaches | The complexity of the target system or software, in terms of both functionality and data, makes students prone to design errors. However, repeated iterations can enhance students' understanding of the target system or software, motivating them to self-correct errors and refine their designs over time. |

## Ⅴ. Summary of Common Phenomena and Feedback from Students

Here, we will systematically summarize the observed common phenomena. First, functional design and data design are the two main aspects of software design. In functional design, when the targeted function involves numerous steps and activities, students tend to overlook some key steps. In data design, students often struggle with distinguishing similar data storage objects and may disregard critical entities or omit essential attributes. These are specific errors commonly made in functional and data design. On the other hand, the following points represent their common behaviors, which are not limited to functional or data aspects. When faced with the same design requirement, students may provide a variety of solutions. Because there is often not a single correct answer in many cases, multiple diverse design outcomes can be considered reasonable. If students have not previously utilized a certain type of information system or software, they may face difficulties in providing comprehensive solutions when tasked with designing related functions and data. The challenge in software design lies in meeting requirements while accurately expressing them according to prescribed model syntax and rules, and for novices, achieving the former is more difficult. The remaining points discuss approaches to assist students in completing their software designs. One approach is to provide students with ready-to-use software for reference, which can simplify their design process. Additionally, teamwork can facilitate the generation of more design ideas and help minimize omissions. It is important to recognize that software design is an iterative process that involves continuous trial and error, and allowing students to engage in multiple iterations and repeated modifications can significantly improve the quality of their design outcomes. The explanations of these common phenomena are listed in Table II.

To investigate the prevalence of common student errors and behaviors identified in our previous multi-case study, as well as to assess the effectiveness of those common supportive approaches, we conducted a questionnaire survey among students after the multi-case study. The participants in this survey consisted of undergraduate students enrolled in the information systems analysis and design and software engineering courses taught by some of the authors in 2024. These students, who hailed from four different classes and totaled 103 individuals, were distinct from the students who participated in the multi-case study conducted in 2023.

The questionnaire employed in this study was straightforward and divided into sections. Firstly, students were asked to indicate whether they had experienced each of the common phenomena during the course. Subsequently, for the common errors and behaviors, students were required to express their agreement on a five-point Likert scale regarding their prevalence. Similarly, for the common supportive approaches, students were prompted to share their views on the effectiveness of these approaches using the same scale. (Please refer to the appendix for the main body of the questionnaire.)

The questionnaire was anonymous and was distributed by the authors during breaks and after class. The authors (also as teachers) first explained the purpose of the survey, provided instructions for filling out the questionnaire, and clarified the meaning of each common phenomenon to the students. Students were then asked to complete the questionnaire during their free time and submit it during the subsequent class. As the questionnaire did not require a significant amount of time from students and was emphasized for serious completion by the teachers, we achieved a 100% response rate.

TABLE III
STATISTICS ON STUDENT FEEDBACK REGARDING THOSE COMMON PHENOMENA

| No. of common phenomena | percentage of students experiencing this | agreement level | |
|---|---|---|---|
| | | mean | standard deviation |
| 1 | 50.49% | 4.02 | 0.88 |
| 2 | 64.08% | 3.56 | 1.20 |
| 3 | 70.87% | 3.54 | 1.10 |
| 4 | 82.52% | 4.27 | 0.92 |
| 5 | 57.28% | 3.61 | 1.07 |
| 6 | 44.66% | 3.05 | 1.05 |
| 7 | 67.96% | 4.14 | 0.97 |
| 8 | 71.84% | 4.20 | 0.94 |
| 9 | 20.39% | 3.82 | 1.00 |

The common phenomenon numbers given in the first column of Table III are exactly the same as those in Table II. The second column of that table shows the proportion of students who report that they have actually experienced these common phenomena in the courses. As can be seen from the table, each common phenomenon has a substantial proportion of students who have experienced it. Fig. 7 further presents the frequency distribution of those common phenomena across types. As shown in the figure, only 0.97% of students report that they have not experienced any common error, which means that 99.03% of students have experienced at least one of the three common errors summarized in this paper. The proportion of students who have experienced two common errors is the highest among all groups, accounting for 41.75%, and 22.33% of students have experienced all three common errors. As for students' common behaviors, no student reports that they have not performed any common behavior, and most students have experienced two of the three common behaviors (accounting for 74.76%). Different from the previous two types of common phenomena, the results of this questionnaire on students' experiences of the common supportive approaches in the courses may not represent the probability of those approaches occurring in normal software design courses. The reason for this is that the teachers of these courses, who participated in the previous multi-case study, already appreciate the practical significance of these approaches. The proportion of students reporting their experiences with these approaches in Table III, and the highest proportion of students experiencing two of those approaches in the courses, as shown in Fig. 7 (60.19%), primarily reflect the proactive promotion of these common supportive approaches by the authors in their own courses.

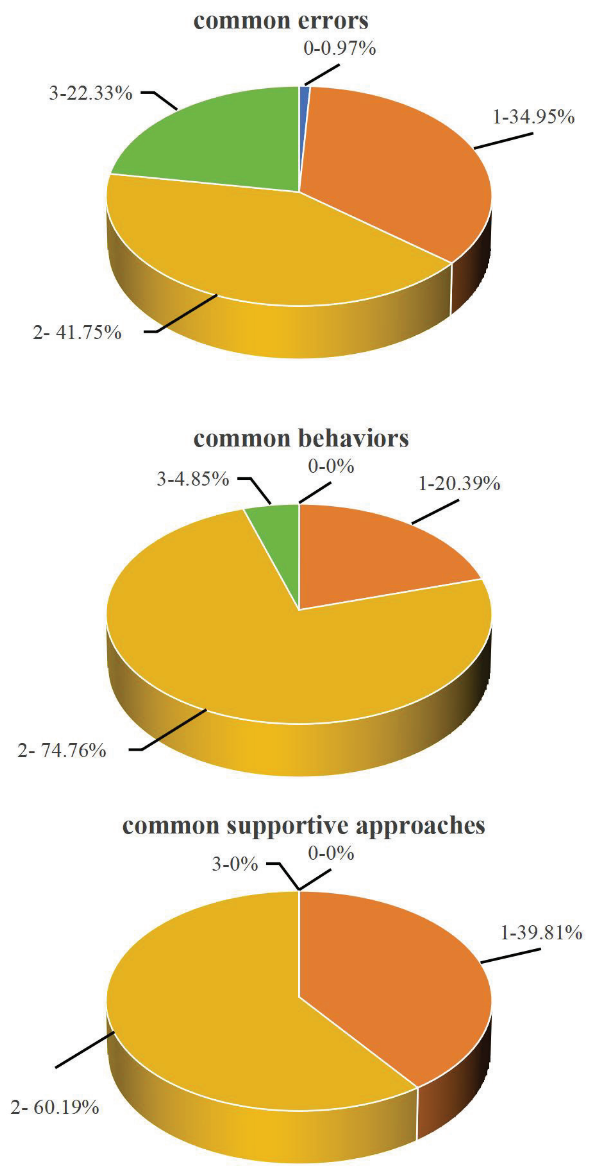


**Fig. 7.** Frequency distribution of common phenomena across types

In contrast, investigating the prevalence of those common supportive approaches seems to be less significant than investigating their effectiveness. Hence, regarding these common supportive approaches, we requested students to evaluate their perceived effectiveness on a five-point Likert scale. For the common errors and behaviors, we measured students' level of agreement regarding their prevalence. The third and fourth columns of Table III present the mean and standard deviation of students' agreement, respectively. From the table, it is evident that students have positively recognized those common phenomena, except for common phenomenon 6, which shows a slightly neutral stance (the mean is 3.05). Regarding phenomenon 6, the highest proportion of students expressed a neutral stance. By discussing the statistical results of certain phenomena with students in classes following the questionnaire survey, it was reported that those who hold a neutral stance on phenomenon 6 generally feel that it is challenging to find an appropriate solution that aligns with design requirements and to express the solution accurately according to prescribed model syntax and rules. They often find it difficult to determine which aspect is more challenging unless prompted to choose between the two, at which point they tend to opt for the former.

When it comes to the effectiveness of those common supportive approaches, the teamwork form of design receives the highest level of student agreement. Many students express a strong desire to complete design tasks in teams during subsequent classroom discussions. Some students who hold negative views on the effectiveness of teamwork mentioned that they often invest more effort but do not receive higher scores compared to other team members.

In summary, the results of the questionnaire survey offer substantial evidence for the prevalence of the common errors and behaviors, as well as the effectiveness of the common supportive approaches outlined in this paper.

## VI. Limitations and Further Research

The main limitation of this paper lies is that the selection of case scenario models, the statistical analysis of students' design outcomes, and the identification of common phenomena are primarily based on a single teaching cycle of the Object-Oriented Information Systems Analysis and Design course. In reality, there is a wide range of available software design models. The chosen models in this paper, namely, TFDs, activity diagrams, sequence diagrams, and ER diagrams, were selected because students demonstrated noteworthy common phenomena in their design processes for these models during a particular teaching cycle. Note that the selection does not imply the superiority of these models over others.

To ensure a certain level of universality, this paper relies on the approval of these common phenomena by four instructors with extensive teaching experience in software design-related courses. However, it is still not possible to completely avoid overlooking some other genuine common phenomena. The next step involves recording students' independent design performances for each model and analyzing their design outcomes in multiple teaching cycles of the course Object-Oriented Information Systems Analysis and Design. This step is aimed at verifying the stability of the statistical results for the selected case scenario models across multiple teaching cycles and at gathering more valuable common phenomena by observing students' design performances on other models. These additional findings will complement the conclusions drawn in this paper.

## VII.CONCLUSION

This paper focuses on the independent design processes of 23 undergraduate students receiving software design education for the same target information system. Based on their behavioral performances and design results in completing TFDs, activity diagrams, sequence diagrams, and ER diagrams, a series of case scenarios are compiled. By analyzing these scenarios, some common phenomena exhibited by those students during the process of receiving software design training are identified. These common phenomena are categorized into common errors, common behaviors, and common supportive approaches. The prevalence of those common errors and behaviors, as well as the effectiveness of those common supportive approaches, was validated through questionnaires distributed to students taking software design courses outside of those participating in the multi-case study.

Here we would like to emphasize the following points: first, the common phenomena among students identified in this paper are closely related to the software design education process and can accurately reflect the distinctive characteristics of software design education. This contribution will assist researchers in understanding the needs and challenges of students in software design education. Second, this paper relies on recording and analyzing common phenomena exhibited by students during the design and presentation phases, combined with a statistical analysis of student design outcomes. Such a research design ensures the objectivity and universality of the findings. Moreover, the analysis of student common behaviors based on their software design outcomes offers a fresh perspective in this field of study. Last, the research environment remains a traditional classroom within educational institutions. Hence, the conclusions drawn in this paper provide valuable guidance for educators engaged in designing and improving classroom experiences for software design courses.

Those common phenomena exhibited by students also pose certain requirements for software educators. When assigning students to design specific software, it is important to consider their prior experience with similar software or functions. For software that students have not previously encountered, providing design references becomes necessary. Given that software design involves a continuous process of trial and error and improvement, students should be encouraged to engage in moderate discussions with their peers and allowed to undergo multiple iterations to gradually enhance their design outcomes. Although teamwork can effectively help integrate students' knowledge to overcome difficulties, reduce errors, and is highly valued by students when completing certain software design tasks, software educators should use this approach cautiously. It is essential to devise a collaborative grading mechanism to prevent any students from exploiting teamwork and to ensure that those who invest more effort receive higher scores. Considering that students may come up with various solutions for the same design task, teachers need to possess excellent software design capabilities to quickly grasp students' design thinking, accurately identify the strengths and weaknesses of their designs, and guide them toward improvement. It is crucial not to blindly dismiss designs that deviate from the expected results, as this may undermine students' enthusiasm and confidence in learning.

## APPENDIX

Main Body of the Questionnaire

| | | During your study of this course, have you had the following experiences? | Do you agree that the following phenomena are prevalent among students during the software design process? 1-strongly disagree, 2-disagree, 3-neutral, 4-agree, 5-strongly agree |
|---|---|---|---|
| 1 | I overlooked key steps or activities when the targeted function involved numerous steps or activities. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 2 | I struggled to distinguish similar data storage objects in data design. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 3 | I disregarded critical entities or omitted essential entity attributes. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 4 | I provided a solution that was completely different from the solutions of the teacher or other classmates. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 5 | I faced difficulties in providing a comprehensive solution when tasked with designing related functions and data, if I had not previously used the target information system or software. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 6 | I faced challenges in envisioning a solution that meets the requirements while accurately expressing it according to prescribed model syntax and rules, with the former being more difficult. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| | | During your study of this course, have you had the following experiences? | Do you agree that the following support approaches are effective? 1-strongly disagree, 2-disagree, 3-neutral, 4-agree, 5-strongly agree |

| | | | |
|---|---|---|---|
| 7 | I referred to some ready-to-use software, which can simplify my design process. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 8 | I adopted teamwork, which can facilitate the generation of more design ideas and help minimize omissions. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |
| 9 | I engaged in multiple iterations and repeated modifications, which can significantly improve their design outcomes. | Yes[ ] No[ ] | 1[ ] 2[ ] 3[ ] 4[ ] 5[ ] |

## REFERENCES

[1] F. Huber and G. Hagel, "Tool-supported teaching of UML diagrams in software engineering education - A systematic literature review," 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2022, pp. 1404-1409.

[2] K. Siau and P. -P. Loo, "Difficulties in Learning UML: A Concept Mapping Analysis," Seventh CAISE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02), Toronto, Canada, 2002, pp. 102–108.

[3] K. Siau and P. -P. Loo, "Identifying difficulties in learning uml," Information Systems Management, vol. 43, 2006, pp. 43-51.

[4] R. Reuter, F. Hauser, C. Gold-veerkamp, J. Mottok, and J. Abke, "Towards a definition and identification of learning obstacles in higher software engineering education," in Proc. 9th International Conference on Education and New Learning Technologies (IATEDEDULEARN), Barcelona, 2017, pp. 10259–10267.

[5] M. Buchenau and J. Fulton Suri, "Experience prototyping. In: Designing interactive systems: processes, practices, methods, and techniques." New York, ACM Press, 2000, pp.424–433.

[6] J. Hu, P. Ross, L. Feijs, and Y. Qian "Uml in action: Integrating formal methods in industrial design education," Technologies for E-Learning and Digital Entertainment: Second International Conference, Edutainment 2007, Hong Kong, China, June 11-13, 2007. Proceedings 2. Springer Berlin Heidelberg, 2007, pp. 489-498.

[7] G. Liebel, O. Badreddin and R. Heldal, "Model Driven Software Engineering in Education: A Multi-Case Study on Perception of Tools and UML," 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), Savannah, GA, USA, 2017, pp. 124-133,

[8] J. Cabot and D. S. Kolovos, "Human factors in the adoption of model-driven engineering: An educator's perspective," in Advances in Conceptual Modeling: ER 2016 Workshops, AHA, MOBID, MORE-BI, MREBA, QMMQ, SCME, and WM2SP, Gifu, Japan, November 14–17, 2016, Proceedings, S. Link and J. C. Trujillo, Eds., 2016, pp. 207–217.

[9] E. Ramollari and D. Dranidis, "Student uml: An educational tool supporting object-oriented analysis and design," Proc. of 11th Panhellenic Conference on Informatics, 2007, pp. 363–373.

[10] G. Liebel, R. Heldal, and J. P. Steghfer, "Impact of the use of industrial modelling tools on modelling education," in 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), 2016, pp. 18–27.

[11] L. Khaled, "A Comparison between UML Tools," in 2009 Second International Conference on Environmental and Computer Science, 2009, pp. 111–114.

[12] H. G. Perez-Gonzalez, A. S. Nunez-Varela, F. E. Martinez-Perez, F. E. Hernandez-Castro, F. Torres-Reyes, R. Juárez-Ramírez, K. Bauer, and C. Guerra-García., "Exploring Software Design Skills of Students in different Stages of their Curriculum," 2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT), Mexico City, Mexico, 2019, pp. 65-71.

[13] H. Zhu, Software design methodology: From principles to architectural styles. Amsterdam, The Netherlands: Elsevier, 2005.

[14] C. Budoya, M. Kissaka, and J. Mtebe, "Instructional design enabled agile method using ADDIE model and feature driven development method," International Journal of Education & Development using ICT, Vol. 15, no. 1, 2019, pp. 35-54.

[15] J. Gal-Ezer, and A. Zeldes, "Teaching software designing skills," Computer Science Education, vol. 10, no.1, 2000, pp. 25-38.

[16] F. P. Brooks, "No Silver Bullet-Essence and accidents of software engineering," IEEE computer, vol.20, no.4, 1987, pp. 10-19.

[17] S. Sircar, S.P. Nerur, and R. Mahapatra, "Revolution or evolution? A comparison of object-oriented and structured systems development methods," MIS Quarterly, vol. 25, no. 4, 2001, pp. 457-471.

[18] G. Booch, "UML in Action," Communications of the ACM, vol. 42, no.10, 1999, pp. 26–28.

[19] J. Surveye, "Java and UML," Available: http://www.devx.com/upload/free/features/javapro/1999/10mid99/js1399/js1399.asp. (1999/2000, Winter).

[20] F. Qiu, L. Zhu, G. Zhang, X. Sheng, M. Ye, Q, Xiang and P Chen, "E-learning performance prediction: Mining the feature space of effective learning behavior," Entropy, vol.24, no.5, 2022, p. 722.

[21]J. Shu, Q. Hu, and M. Zhi, "Research on the Learning Behavior of University Students in Blended Teaching," International Journal of Information and Education Technology, vol. 9, no. 2, 2019, pp. 92-98.

[22] S. Berengarten, "Identifying learning patterns of individual students: An exploratory study." Social Service Review, vol.31, no.4, 1957, pp407-417.

[23] D. Darby, "An intensive study unit in operative dentistry: a first study," Doctor degree dissertation, the Department of Educational Psychology, The state University of Iowa, Iowa, USA, 1964.

[24] P. Dillon, R. Wang, and P. Tearle, “Cultural disconnection in virtual education,”Pedagogy, Culture & Society, vol.15, no.2, 2007, pp153–174.

[25] Z. Wang, A. Qadir, A. Asmat, M. Aslam Mian, X. Luo, “The Advent of Coronavirus Disease 2019 and the Impact of Mobile Learning on Student Learning Performance: The Mediating Role of Student Learning Behavior,” Frontiers in psychology, vol.12, 2022, p796298.

[26] R. Johar, and M. Ramli, “Mathematical model of student learning behavior with the effect of learning motivation and student social interaction,” Journal on Mathematics Education, vol.13, no.3, 2022, pp415-436.

[27] T. Raupach, J. Brown, S. Anders, G. Hasenfuss, and S. Harendza, “Summative assessments are more powerful drivers of student learning than resource intensive teaching formats,” BMC medicine, vol.11, 2013, pp1-10.

[28] A. Roth, S. Ogrin, and B. Schmitz, “Assessing self-regulated learning in higher education: A systematic literature review of self-report instruments,” Educational Assessment, Evaluation and Accountability, vol.28, 2016, pp225-250.

[29] E. Fincham, D. Gašević, J. Jovanović and A. Pardo, "From Study Tactics to Learning Strategies: An Analytical Method for Extracting Interpretable Representations," IEEE Transactions on Learning Technologies, vol. 12, no. 1, 2018, pp. 59-72.

[30] S. A. Priyambada, M. Er, B. N. Yahya and T. Usagawa, “Profile-Based Cluster Evolution Analysis: Identification of Migration Patterns for Understanding Student Learning Behavior,” IEEE Access, vol. 9, 2021, pp. 101718-101728.

[31] J. Wong, M. Baars, D. Davis, T. Van Der Zee, G. Houben, and F. Paas, “Supporting self-regulated learning in online learning environments and MOOCs: A systematic review,” International Journal of Human Computer Interaction, vol. 35, no. (4–5), 2019, pp. 356–373.

[32] D. Zhidkikh, M. Saarela, and T. Kärkkäinen, “Measuring self-regulated learning in a junior high school mathematics classroom: Combining aptitude and event measures in digital learning materials,” Journal of Computer Assisted Learning, 2003, Available:https://onlinelibrary.wiley.com/doi/full/10.1111/jcal.12842 .

[33] B.K. Keogh and L.D. Becker, “Early detection of learning problems: Questions,cautions, and guidelines,” Exceptional Children, vol.40, no.1, 1973, pp5–11.

[34] R. Nambiar, N. Nor, K. Ismail, and S. Adam, “New Learning Spaces and Transformations in Teacher Pedagogy and Student Learning Behavior in the Language Learning Classroom,” 3L: The Southeast Asian Journal of English Language Studies, vol.23, no.4, pp29 – 40.

[35] N. W. Tsai, “Assessment of students’ learning behavior and academic misconduct in a student-pulled online learning and student-governed testing environment: A case study,” Journal of Education for Business, vol.91, no. 7, 2016, pp387-392.

[36] Y. Wang, “A Research on the Mixed Teaching Mode of "MATLAB Simulation Technology" Based on Based on MOOC+SPOC+Flipped Classroom,” 2021 3rd International Conference on Computer Science and Technologies in Education (CSTE), Beijing, China, 2021, pp. 9-13.

[37] B. Lu, C. Liu, and T. Zhao, "A three-tier salary management system for higher vocational colleges," International Journal of Multimedia and Ubiquitous Engineering, vol.10, no.4, 2015, pp. 91-104.

[38] Y. Wang, "Realization of Automobile Marketing Management System Based on. NET Structure," Iberian Journal of Information Systems and Technologies, no.18A, 2016, pp. 125-140.

[39] J. Chen, R. Gu, C. Li, Information Systems Analysis and Design Tutorial, Beijing, China: Posts & Telecom Press, 2010.

[40] J. Chen, R. Gu, B. Xu, Information Systems Development Methodology Tutorial (5th Edition), Beijing, China: Tsinghua University Press, 2019.

[41] A. Nayak, and S. Debasis, "Synthesis of test scenarios using UML activity diagrams," Software & Systems Modeling, vol. 10, no. 1, 2011, pp. 63-89.

[42] S. Herrmann, C. Hundt, K. Mehner, "Mapping use case level aspects to ObjectTeams/Java," In A. Moreira et. al, editor, OOPSLA Workshop on Early Aspects, 2004.

[43] J. Pancham, “Determining and developing appropriate methods for requirements verification and modelling of telecentre operational monitoring in a developing country,” Master degree dissertation, Faculty of Accounting and Informatics, Durban University of Technology, Durban, South Africa, 2017.

[44] M. Jaragh and K. A. Saleh, "Modeling communications protocols using the Unified Modeling Language," 2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No.00CH37119), Kuala Lumpur, Malaysia, vol.2, 2000, pp. 271-275.

[45] Z. An, and D. Peters, "On the Description of Communications Between Software Components with UML," Proceedings of Newfoundland Electrical and Computer Engineering Conference, St. John's, Newfoundland, Canada, 2003.

[46] H. Wang, T. Feng, J. Zhang and K. Zhang, "Consistency check between behaviour models," IEEE International Symposium on Communications and Information Technology, 2005. ISCIT 2005., Beijing, 2005, pp. 486-489.

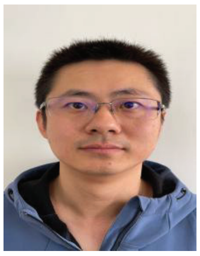
[47] H. Saiedian, "An evaluation of extended entity-relationship model," Information and software Technology, vol. 39, no.7,1997, pp. 449-462.

[48] C. Batini, S. Ceri, and S.B. Navathe, Conceptual Database Design - An Entity Relationship Approach, Redwood City, CA, USA: Benjamin-Cummings Publishing Company, 1992 .

[49] T.A. Bruce, Designing quality databases with IDEF1X information models, New York, NY, USA: Dorset House Publishing Company, 1992.

[50] D.A. Koonce, and R.P. Judd "A visual modelling language for EXPRESS schema," International Journal of Computer Integrated Manufacturing, vol.14, no.5, 2001, pp. 457-472.

**Tao Fu** received his Ph.D in Management Science and Engineering from Beijing University of Technology in 2011. He is currently an associate professor and a Master Tutor of Economics and Management School, Beijing University of Technology. His main research interests include Complex Network, Social Network and Multi-Agent Simulation. As the instructor, he has taught the course Object-Oriented Information System Analysis and Design continuously for nearly 15 years.



**Ran Sun** received his Master's degree in Software Engineering from Beijing University of Technology in 2015. He is currently a Senior Engineer of Computer School, North China Institute of Aerospace Engineering. His main research interests include Software Engineering, Complex Network, and Computer Simulation. As the instructor, he has taught the course Software Engineering many cycles.

**Chenguang Li** received his Ph.D in Management Science and Engineering from Beijing University of Technology in 2014. He is currently an associate professor and a Master Tutor of Economics and Management School, North China University of Technology. His main research interests include Economic Policy, Complex Network, and Multi-Agent Simulation. As the instructor, he has taught the course Management Information System continuously for nearly 10 years.

**Lian Wang** received her Ph.D in Management Science and Engineering from Central University of Finance and Economics in 2010. She is currently a Lecturer of Economics and Management School, Beijing University of Technology. Her main research interests include Information System Analysis and Design, Financial Management and Software Engineering. As the instructor, she has taught the course Object-Oriented Information System Analysis and Design many cycles.

CONTENTS PAGE