Journal of Information Systems Education

Volume 31
Issue 4
Fall 2020

Constructive Use of Errors in Teaching the UML Class Diagram in an IS Engineering Course

Ronit Shmallo and Tammar Shrot

Recommended Citation: Shmallo, R. & Shrot, T. (2020). Constructive Use of Errors in Teaching the UML Class Diagram in an IS Engineering Course. *Journal of Information Systems Education,* 31(4), 282-293.

Article Link: http://jise.org/Volume31/n4/JISEv31n4p282.html

Initial Submission:	21 October 2019
Accepted:	17 March 2020
Abstract Posted Online:	8 September 2020
Published:	10 December 2020

ISSN: 2574-3872 (Online) 1055-3096 (Print)

Constructive Use of Errors in Teaching the UML Class Diagram in an IS Engineering Course

Ronit Shmallo
Industrial Engineering and Management Department
Shamoon College of Engineering
Ashdod, Israel
ronits1@sce.ac.il

Tammar Shrot
Software Engineering Department
Shamoon College of Engineering
Ashdod, Israel
tammash@sce.ac.il

ABSTRACT

A class diagram is one of the most important diagrams of Unified Modeling Language (UML) and can be used for modeling the static structure of a software system. Learning from errors is a teaching approach based on the assumption that errors can promote learning. We applied a constructive approach of using errors in designing a UML class diagram in order to (a) categorize the students' errors when they design a class diagram from a text scenario that describes a specific organization and (b) determine whether the learning-from-errors approach enables students to produce more accurate and correct diagrams. The research was conducted with college students (N = 45) studying for their bachelor's degree in engineering. The approach is presented, and the learning-from-errors activity is illustrated. We present the students' errors in designing the class diagram before and after the activity, together with the students' opinions about applying the new approach in their course. Twenty errors in fundamental components of the class diagram design were observed. The students erred less after the activity of learning from errors. The displayed results show the relevance and potential of embedding our approach in teaching. Furthermore, the students viewed the learning-from-errors activity favorably. Thus, one of the benefits of our developed activity is increased student motivation. In light of the improved performance of the task, and the students' responses to the learning-from-errors approach, we recommend that information systems teachers use similar activities in different fields and on various topics.

Keywords: IS education, Object-oriented modeling, Unified modeling language (UML), Active learning, Peer evaluation

1. INTRODUCTION

Unified Modeling Language (UML) is the standard formalism for the object-oriented analysis and design of software systems (Berardi, Calvanese, and De Giacomo, 2005), and it provides a graphical representation of the analysis and design of the system models (Dranidis, Stamatopoulou, and Ntika, 2015; Störrle, 2017). Class diagrams are perceived as one of the most important components of UML, mapping out the static structure of a system by modeling its classes, attributes, methods, and the relationships between classes. It is critical that the class diagrams present information clearly, but the reasoning of UML class diagrams can be a complex task for novice designers (Berardi, Calvanese, and De Giacomo, 2005).

Learning how to model and create class diagrams has become a necessity for information systems (IS) students and a substantial challenge arising before IS educators. Novice designers tend to make errors when learning to design models, and it is important to address the errors at a very early stage of a modeler's education (Bogdanova and Snoeck, 2018). When one is designing a system in the real world, it is crucial to detect errors as early as possible in the development process to reduce later development costs (Cabot, Clarisó, and Riera, 2014).

Everyone errs; however, learning from errors has been proven to be as profitable as, if not more than, learning the correct form to begin with (Ginat and Shmallo, 2013). The learning-from-errors approach involves an aspect of constructivism since the utilization of errors is based on students' prior knowledge which may be inaccurate or faulty. Therefore, learning from errors can be useful if there is a constructivist construction. The review of one's knowledge and procedures involves a metacognitive aspect that refers to one's knowledge concerning one's own cognitive processes or the learning of relevant properties of information or data (Flavell,

1976). The improvement of one's metacognitive skill is an additional constructive means for learning (Schoenfeld, 2009). It appears that using errors in a structured learning process, followed by feedback, discussion, and correction, leads to a better understanding (Tulis, Steuer, and Dresel, 2016; Metcalfe, 2017). Engaging with errors is difficult, but the difficulty can be a powerful means for learning (Borasi, 1996; Bjork, 2012) if the errors are used to scaffold the construction of learner knowledge. Engagement with errors also has the benefits of deep discussion of thought processes and a learning environment that challenges students to be actively involved in the learning process (de Freitas, Silva, and Marsicano, 2016).

This paper presents a study of constructive learning from errors in a theoretical IS course in engineering studies. The novelty of this paper is the implementation of the learning-from-errors approach in an IS engineering teaching process. The approach was applied in the analysis and design of an IS course of 45 undergraduate students. The main goal was to improve the learning process by integrating the learning-from-errors approach into the process of studying class diagrams. We examined whether this integration enabled students to better understand UML class diagrams and improve their designs to make them correct and more accurate.

2. BACKGROUND

2.1 UML Class Diagrams

UML is a standard language for modeling object-oriented systems that is commonly used in the software industry (Silva, Steinmacher, and Conte, 2017). Object-oriented models are based on objects, and objects are instances of classes. Classes have properties, behavior (functionality), and relationships of several kinds with other classes. The object-oriented model can be presented as a UML class diagram, which displays in one picture the set of classes and a set of relationships between them, such as inheritance (generalization), association, aggregation, and composition.

Although class diagrams are only one of the diagram types included in UML, research has shown that they are perceived as the most important (e.g., Erickson and Siau, 2004). Changing system requirements causes changes in the structure of the system, and they need to be reflected in modifications of the class diagrams. Hence, it is important to develop diagrams that can easily incorporate changes (Genero, Piattini, and Calero, 2000).

UML class diagrams for modeling software systems can be large and complex in the design, analysis, and maintenance stages. Students face difficulties while learning how to model complete and correct class diagrams and that can affect the final software quality, since these diagrams would represent the software incompletely and incorrectly (Siau and Loo, 2006; Szmurło and Śmiałek, 2006; Lethbridge, 2014).

One of the support options in the design of the class diagrams phase is to use an intelligent tutoring system or computer-aided software engineering (CASE) tools. Educators use a variety of educational tools in order to help their students with conceptual modeling software systems, but most of those tools are rather small and limited, with a small subset of UML features (Dranidis, Stamatopoulou, and Ntika, 2015). For example, Baghaei, Mitrovic, and Irwin (2007) presented a tutor that teaches how to design UML class diagrams and how to provide feedback on collaboration using the same formalism. Gutwenger et al. (2003) suggested a new approach for visualizing UML class diagrams leading to a balanced mixture of some aesthetic criteria. Ramollari and Dranidis (2007) developed StudentUML as a modeling tool that supports consistency checking of all the UML taught diagrams, including class diagrams. Cabot, Clarisó, and Riera (2014) presented methods for the verification of UML class diagrams extended with Object Constraint Language (OCL) constraints. In addition, CASE tools are commonly adopted for student use: power designer (https://sybase-powerdesigner.apponic.com), visible analyst (https://dbmstools.com/tools/visible-analyst), or visual paradigm (https://www.visual-paradigm.com). However, none of those tutoring systems or support tools can prevent novice designers from making various forms of inaccuracies, omissions, or redundancies in their class diagram designs.

2.2 Learning-from-Errors Approach

The literature describes a variety of errors that beginners tend to run into, but it rarely offers solutions that deal with these errors. In many cases, the suggested methods for overcoming errors are specific to a single scope, addressing one instance of an error (e.g., Watson, 2006; Sanders and Thomas, 2007; Booth et al., 2013; Casterella and Vijayasarathy, 2019). The cause of an error is an erroneous perception, which frequently derives from overly general knowledge structures or from vague, faulty, or missing knowledge components (Ohlsson, 1996). Errors are experienced as conflicts between the knowledge that the learner believes to be correct and what the learner perceives as the present situation. This knowledge gap should be closed by error-detection and error-correction (Mathan and Koedinger, 2005). Error-detection will reveal gaps by comparing actual with expected outcomes. Error-correction will close the gaps by specializing faulty knowledge structures so that they become active only in situations in which they are appropriate. The cognitive conflict will stimulate the learner's process of reflection and critical thinking (Borasi, 1996), which will lead the student to understand the source of the error. This in turn will lead to a revision of the learner's knowledge (Ohlsson, 1996).

The constructivist view of how knowledge is attained has important implications for an educational approach to errors (Borasi, 1996). Constructive use of errors as a teaching approach is based on students' prior knowledge, which is inaccurate or somewhat vague. Learners cannot progress in learning if they do not have the relevant mental-model modification at which the instruction was aimed (Ben-Ari, 1998). The goal is to improve students' knowledge and skills by creating cognitive conflict through errors. Constructivist studies support the notion of conflict as a catalyst for learning (e.g., Borasi, 1996; Ginat and Shmallo, 2013). Conflict causes examination of learner knowledge and procedures (Confrey, 1990) and leads to a deeper conceptual understanding and greater awareness of the errors to be avoided.

In the past two decades, researchers have presented convincing evidence of the benefits of the learning-from-errors approach in different domains, such as mathematics, physics, and computer science (e.g., Borasi, 1996; Pinkerton, 2005; Yerushalmi and Polingher, 2006; Ginat and Shmallo, 2013). Learning from errors can promote learning, and errors can be used as a motivator for learning (e.g., Borasi, 1994; Siegler, 2002; Curry, 2004; Große and Renkl, 2007; Siegler and Chen,

2008; Ginat and Shmallo, 2013; Bogdanova and Snoeck, 2018).

There is a growing consensus that students can learn effectively from their errors. However, textbooks do not usually include incorrect examples, and creating materials that include incorrect examples can be time consuming for teachers (Booth, Begolli, and McCann, 2016). Furthermore, teachers prefer to avoid talking about errors in class because they are afraid their students will adopt the errors in their own problem-solving (Santagata, 2004).

Researchers have shown the benefits of using the learning-from-errors approach in various activities. For example, some offer supportive visual environments (Melis, Sander, and Tsovaltzi, 2010). Others have students think about and correct their own errors (Henderson and Harper, 2009; Cherepinsky, 2011); some use erroneous statements that the students are asked to analyze and diagnose (Yerushalmi and Polingher, 2006); some integrate incorrect examples into class assignments (Booth et al., 2013); and some use self-explanations of both correct and incorrect solutions (Siegler, 2002; Curry, 2004; Große and Renkl, 2007).

To the best of our knowledge, no researchers have conducted studies on the learning-from-errors approach in IS except Bogdanova and Snoeck (2018). They examined a primarily learning-from-errors activity, with master's students, to build a simple UML model based on textual case description. Their aim was to teach conceptual modeling by identifying the most common errors in students' models and then introducing error-based step-by-step exercises. Error detection related only to class-level errors and association-level errors. Their primary conclusion was that the step-by-step exercises were effective, at least when it concerns the immediately following exercise.

3. RESEARCH DESCRIPTION

3.1 Research Rationale and Questions

Numerous studies have been conducted over the years on techniques for modeling class diagrams and ways to improve the students' designs (e.g., Berardi, Calvanese, and De Giacomo, 2005; Bock and Yager, 2005; Carte, Jasperson, and Cornelius, 2006; Watson, 2006; Queralt and Teniente, 2012; Cabot, Clarisó, and Reira, 2014). The rationale for our research was to suggest a new approach for teaching novice designers how to design a correct and accurate UML class diagram. The research had two intentions. The first was to map and catalog errors that students make in modeling the class diagrams of their chosen organization. The second was to examine a new teaching method that integrates explicit orientation toward errors and striving to learn from them. We wished to examine whether using the learning-from-errors approach improved students' class diagram designs. The research questions were the following:

- What errors do students make when they are designing class diagrams?
- Do students improve their class diagrams as a result of using the learning-from-errors approach?
- Do students like the learning-from-errors approach?
- Do students believe it advanced their knowledge and understanding of class diagrams?

3.2 Research Participants

The research was conducted with 45 students studying for an undergraduate degree in engineering in the Department of Industrial Engineering and Management (IEM). The course was the Analysis and Design of Information Systems.

3.3 Research Tool and Methods

The research applied a mixed-method approach combining qualitative and quantitative methods (Creswell and Creswell, 2017). The study used quantitative analysis and qualitative research methods to expand and complement the quantitative findings. The research tools were the learning-from-errors activity and an attitudes questionnaire about the learning-from-errors approach.

3.3.1 Learning-from-errors activity. The goal was to design a class diagram for an organization that the students had chosen for the final project in their degree. Each student pair and one student without a partner selected a specific organization, such as an information system for students interested in higher education, a system for assigning replacement teachers in schools, an information system for the management of online purchasing groups, or a system for a charity organization. The activity had two steps: (1) designing a class diagram of the chosen organization according to a text scenario that described the organization and its main process and (2) refining the diagrams from the first step by using the approach of learning from errors (a detailed description about the activity appears in section 3.5).

3.3.2 Attitudes questionnaire. A questionnaire was distributed in the classroom. It asked students their opinion on aspects of the learning-from-errors approach that they encountered during their class diagram activity.

3.4 Analysis and Design of Information Systems Course

The course Analysis and Design of Information Systems is taught in the IEM department and is part of the IS track. This is a required course in the 6th semester. The prerequisite for this course is the Object-Oriented Programming (OOP) course.

The Analysis and Design of Information Systems course familiarizes students with the methodologies, tools, and methods for developing a software or information system. The course focuses on systems developed on the basis of OOP paradigms and is based on the book *Systems Analysis & Design: An Object-Oriented Approach with UML* (Dennis, Wixom, and Tegarden, 2015).

System development is introduced using the UML tools in general, while focusing on the main diagrams: use case, class, activity, state machine, and sequence. Information about those UML tools and how to build them is based on *Guide to Applying the UML* (Alhir, 2006).

The learning outcomes of the course are that upon successful completion, the students will be able to do the following:

1. Use an object-oriented approach to describe and implement the stages in developing software or information systems.
2. Define user functional and non-functional requirements.
3. Create use case and activity diagrams to define user functional requirements.

4. Use class diagrams to design the system components.
5. Use sequence diagrams to model the functionality of a system.
6. Use activity and state machine (state-chart) diagrams to describe the dynamic aspects of, and flows in, the system.
7. Integrate various diagrams in order to analyze and design a full system.

This paper focuses on the fourth target of the course: students using class diagrams to design the components of their chosen organization.

3.5 Process

The learning-from-errors activity was divided into five stages:

1. During 2 weeks of the 14-week course, the students attended a weekly 3-hour lecture on how to design system components using class diagrams. The students were given a task of describing their chosen organization (a text scenario) and designing a class diagram accordingly. They were divided into pairs and given 2 weeks to submit their solutions.
2. The teacher sent a personal e-mail to each pair with another pair's task solution and the description of their organization. Each pair was asked to examine the solution they received and to mark any errors they found in their colleagues' solution regarding the presented organization. Students were expected to submit this second assignment two weeks later (the "Before" stage).
3. Course staff sent each pair a file containing their colleagues' comments. Students were told to carefully examine the comments they received from their colleagues since there was no guarantee that the comments were correct.
4. The students were asked to submit a revised class diagram design in two weeks, and the course staff graded this final assignment (only). The course staff analyzed students' solutions after stages 2 and 4. The solutions were compared to identify the errors that the students found in their colleagues' solutions and to discover whether the refined class diagram, which was submitted after the learning-from-errors activity, had gained accuracy. Emphasis was placed on errors in the following fundamental components of the class diagram design: identifying the keys, attributes, classes; the class relationships type (aggregation, inheritance, associations); the names of relationships (written in the middle of the association line); and multiplicity (how many objects of each class take part in the relationships). The points that the course staff did not check for errors were visibility of class attributes and methods, the class notation, and abstract classes (the "After" stage).
5. Finally, the teacher initiated a classroom discussion of typical errors that frequently appeared in students' solutions. The discussion was general, since each pair had based their diagram on a different organization. However, the teacher emphasized the common errors, such as how to recognize an aggregation relation, an inheritance relation, a primary key, and an attribute of a class. At the end of the course, the students submitted a final version of all the UML diagrams of their system, including the class diagram.

4. RESULTS

4.1 Study

Twenty-three solutions were analyzed from 45 students divided into 22 pairs and 1 sole student. The solutions included students' written answers about the errors they found in their colleagues' class diagram designs after stage 2. The analysis determined which components the students emphasized when they were searching for errors in their colleagues' solutions, as can be seen in Table 1. We classified the error types according to those components while counting the number of students that erred in each type, as presented in Table 1 ("Before the Activity" column). We also analyzed the students' errors in the class diagram design after stage 4 in order to compare the two stages and to see if there was improvement after the applied approach. We did not search for new errors after stage 4. The students' errors that were found after stage 4 can be seen in Table 1 ("After the Activity" column).

Shmallo, Ragonis, and Ginat (2012) used a dichotomous categorization of expansion–reduction to classify the error types made by novice programmers learning OOP concepts. Katz and Shmallo (2015) used the terms *addition/omission* and classified novice designers' errors in understanding the conceptual modeling of relational databases. They found that the most common errors made by students are the addition type, but errors of omission are also quite prevalent.

On the basis of those classifications, we also classified the errors that we found in the basic components of the class diagram designs according to a dichotomous categorization of addition/omission, as presented in Table 1. The purpose was to check if educators can get an additional perspective on common errors that novice designers make when they design class diagrams.

All errors made by all pairs were counted. When a pair was wrong in several component types, their errors were counted separately for each type. However, if a pair erred several times in the same error type, their errors were counted only once. After analyzing the students' errors, we divided the errors into four basic components of the class diagram: keys (3 errors), attributes (3 errors), classes (4 errors), and relations (10 errors). Most of the errors deal with the relations components. Students mostly err in identifying relations between classes. The most common error, before and after the activity, was that "the relation specified between classes was incorrect or unnecessary (and it is not an aggregation or inheritance relation)." A high percentage of students also erred in "identifying the relation of aggregation between classes."

From the frequency of errors appearing in Table 1, we noticed a total of 60 of the omission type (58%) versus 44 errors of the addition type (42%) from the first stage, and those proportions switched in the second stage to 23 (43%) versus 31 (57%), respectively. Over the two stages, students made slightly more omission errors (53%) than addition errors (47%), but the difference appeared insignificant ($p > 0.05$), consistent with the non-significant differences between the two types in each stage. This means that we cannot argue for more errors of one type over the other.

Component	Error	Before the Activity (%)	After the Activity (%)	Addition/ Omission
Keys	The definition of a primary key is not unique.	17.391	8.696	Addition
	A primary key is defined in existence dependency instead of a partial key.	13.043	13.043	Addition
	The marking of a primary key is lacking.	21.739	4.348	Omission
Attributes	Attributes described in the text scenario are declared in wrong class.	8.696	13.043	Addition
	Attributes described in the text scenario are missing from the class diagram.	34.784	8.696	Omission
	Unnecessary attributes that appear in the class diagram were not mentioned in the text scenario.	26.087	13.043	Addition
Classes	There is an unnecessary class in the diagram.	13.043	8.696	Addition
	There is an unnecessary associative class in the diagram.	4.348	4.348	Addition
	There is a missing class in the class diagram.	8.696	0	Omission
	There is a missing associative class in the class diagram.	13.043	4.348	Omission
Relations	The diagram is lacking an aggregation relation between classes.	60.870	26.087	Omission
	The diagram is lacking an inheritance relation between classes.	17.391	4.348	Omission
	There is an unnecessary relation in the class diagram that was not mentioned in the scenario.	21.739	17.391	Addition
	Relation described in the scenario is missing from the class diagram.	30.435	21.739	Omission
	The aggregation or inheritance relation between two classes is wrong.	13.043	8.696	Addition
	The diagram lacks marking of the role of the relation type between classes.	21.739	8.696	Omission
	A relation specified between two classes is incorrect or unnecessary (and it is not an aggregation or inheritance relation).	73.913	47.826	Addition
	The diagram is missing relation type between two classes.	26.087	13.043	Omission
	The diagram is missing the multiplicity between two classes.	26.087	8.696	Omission
	The diagram is lacking marking of X (eXclusive) and T (Total cover) in an inheritance relation between classes.	13.043	0	Omission

Table 1. Errors Made by Students and Their Frequency (N = 23 Solutions)

4.2 Modeling Strategy and Methods

This study aims at assessing the potential of the learning-from-errors approach by comparing individuals' errors over time (before vs. after the learning-from-errors activity). The study is subject to our hypothesis that teaching students how to design class diagrams using learning from errors is expected to reduce the individual's subsequent level of errors. To test this hypothesis, we use a series of two-level logistic regression models that look at the probability of making each type of error before versus after the learning activity. In these logistic models we use the generalized estimating equations (GEE) approach, which allows for repeatedly measured individuals; that is, each individual student is measured twice, and correlations within individuals are assumed. The GEE approach takes on various distribution types, among which is logistic distribution (logit model). This model generates marginal probabilities for making the error before and after the constructive learning and compares across them (Hardin and Hilbe, 2012). For the overall level of error, we use the generalized linear mixed model (GLMM) with log link and Poisson distribution, where the Poisson distribution is a unique, discrete distribution with the mean equal to the variance (Hilbe, 2017). Put differently, the total number of errors is a count variable that receives integer, non-negative values and has a right-hand tail of small frequencies.

In Table 2, we show the frequency of making an error across the 20 types of possible errors examined. Clearly, there is a reduction in the total number of errors from the preliminary test to the subsequent test ($F(1,22) = 65.75$, $p < 0.001$). Overall, from almost 5 out of 20 possible errors before the learning, the mean number is reduced to only 2.35 out of 20 after the learning. This change in the total distribution of errors is illustrated in Figure 1.

No. of Errors	Before		After		*F* Test
	Freq.	%	Freq.	%	$F(1,22), p, \eta_p^2$
0	0	0	1	4.3	65.74, < 0.001, 0.749
1	0	0	1	4.3	
2	0	0	11	47.8	
3	3	13.0	9	39.1	
4	8	34.8	1	4.3	
5	7	30.4	0	0	
6	4	17.4	0	0	
7	1	4.3	0	0	
Total *N*	23	100.0	23	100.0	
No. of errors	107		54		
Mean	4.65		2.35		
SE	0.22		0.17		

Table 2. Frequency of Errors by Time

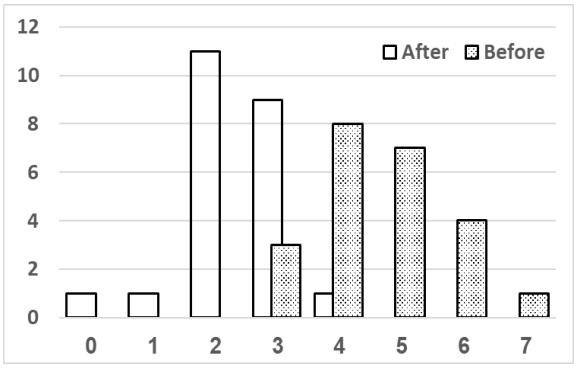


Figure 1. Number of Errors Made Before and After the Learning Activity

Note: Horizontal axis shows frequencies across the 23 solutions; vertical axis, number of error.

We can also see from Figure 1 that at least the after-learning distribution follows the Poisson distribution, with one student making no errors at all and half of them making one or two errors, whereas before the learning each student made at least three errors.

A finer analysis that looked at each type of error is presented in Table 3. The table shows the time effect as estimated by the GEE procedure with correlated response.

The reduction in the probability of making a certain error over time appears to be significant for error "Attributes described in the text scenario are declared in wrong class" (Attributes error) ($b = -1.72$, $p = 0.012$); error "The diagram is lacking an aggregation relation between classes" (Relations error) ($b = -1.48$, $p = 0.001$); error "A relation specified between two classes is incorrect or unnecessary (and it is not an aggregation or inheritance relation)" (Relations error) ($b = -1.13$, $p = 0.007$); and error "The diagram is missing the multiplicity between two classes" (Relations error) ($b = -1.31$, $p = 0.040$). Although this trend is limited to those errors, the marginal probabilities show that at the "Before" stage the predicted probability of making the error is higher in comparison to the predicted probability at the "After" stage, except for error "A primary key is defined in existence dependency instead of a partial key" (Keys error), which shows equal marginal probabilities at the "Before" and "After" stages. In error "There is a missing class in the class diagram" (Classes error) and error "A primary key is defined in existence dependency instead of a partial key" (Keys error), no comparison is done, as no errors of these types are made among the 23 student pairs in the experiment. Finally, a GLMM Poisson regression results in a significant time effect that follows the actual reduction in the number of errors ($b = -0.068$, $p = 0.003$). The model predicts 4.65 errors before learning but only 2.35 after learning; this reduction is illustrated in Figure 2, which also shows the upper and lower 95% confidence interval.

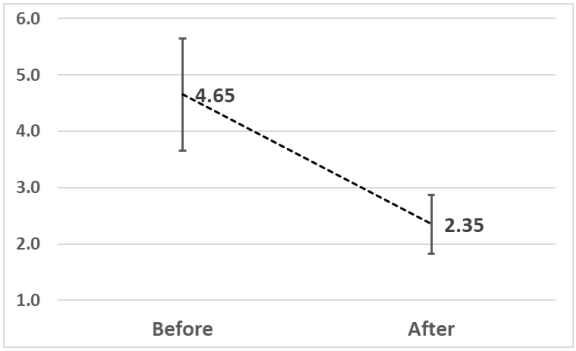


Figure 2. Expected Level of Errors Over Time; Means and Lower and Upper Bounds (95% Confidence)

Note: Horizontal axis shows time of measurement; vertical axis, number of errors.

In addition, students were asked to fill out a position questionnaire that contained statements concerning their assessments throughout the learning-from-errors activity. Students had to choose a value on a scale of 1 to 4, where 1 represented "strongly disagree," 2 "disagree," 3 "agree," and 4 "strongly agree." The results can be seen in Figure 3.

Error	Before	After	Time Effect Coefficient (SE)	Sig. *p*-Value	Wald Chi-Square
The definition of a primary key is not unique	0.17 (0.08)	0.09 (0.06)	-0.79 (0.55)	0.149	2.08
A primary key is defined in existence dependency instead of a partial key	0.13 (0.07)	0.13 (0.07)	0.00 (0.54)	1.00	0.00
The marking of a primary key is lacking	0.22 (0.09)	0.04 (0.04)	-1.81 (0.94)	0.054*	3.71
Attributes described in the text scenario are declared in wrong class	0.09 (0.06)	0.13 (0.07)	0.45 (0.79)	0.564	0.33
Attributes described in the text scenario are missing from the class diagram	0.35 (0.10)	0.09 (0.06)	-1.72 (0.68)	0.012**	6.38
Unnecessary attributes that appear in the class diagram were not mentioned in the text scenario	0.26 (0.09)	0.13 (0.07)	-0.86 (0.47)	0.072*	3.25
There is an unnecessary class in the diagram	0.13 (0.07)	0.09 (0.06)	-0.45 (0.45)	0.311	1.03
There is an unnecessary associative class in the diagram	0.04 (0.04)	0.04 (0.04)	0.00 (0.00)	1.00	0.00
There is a missing class in the class diagram	0.09 (0.06)	0.00 (0.00)	—	—	—
There is a missing associative class in the class diagram	0.13 (0.07)	0.04 (0.04)	-1.19 (0.86)	0.163	1.95
The diagram is lacking an aggregation relation between classes	0.61 (0.10)	0.26 (0.09)	-1.48 (0.46)	0.001***	10.25
The diagram is lacking an inheritance relation between classes	0.17 (0.08)	0.04 (0.04)	-1.53 (0.91)	0.092*	2.85
There is an unnecessary relation in the class diagram that was not mentioned in the scenario	0.22 (0.09)	0.17 (0.08)	-0.28 (0.48)	0.562	0.34
Relation described in the scenario is missing from the class diagram	0.30 (0.10)	0.22 (0.09)	-0.45 (0.31)	0.142	2.15
The aggregation or inheritance relation between two classes is wrong	0.13 (0.07)	0.09 (0.06)	-0.45 (0.45)	0.311	1.03
The diagram lacks marking of the role of the relation type between classes	0.22 (0.09)	0.09 (0.06)	-1.07 (0.60)	0.076*	3.14
A relation specified between two classes is incorrect or unnecessary (and it is not an aggregation or inheritance relation)	0.74 (0.09)	0.48 (0.10)	-1.13 (0.42)	0.007***	7.31
The diagram is missing relation type between two classes	0.26 (0.09)	0.13 (0.07)	-0.86 (0.47)	0.072*	3.25
The diagram is missing the multiplicity between two classes	0.26 (0.09)	0.09 (0.06)	-1.31 (0.64)	0.040**	4.21
The diagram is lacking marking of X (eXclusive) and T (Total cover) in an inheritance relation between classes	0.13 (0.07)	0.00 (0.00)	—	—	—
Overall	4.65 (1.00)	2.35 (0.52)	-0.68 (0.22)	0.003***	$t = 3.12$

Note: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$; "Before" and "After" indicate marginal probabilities of making the error of that type; standard errors in parentheses.

Table 3. Error-Type Comparisons Over Time: Results of Generalized Estimating Equations

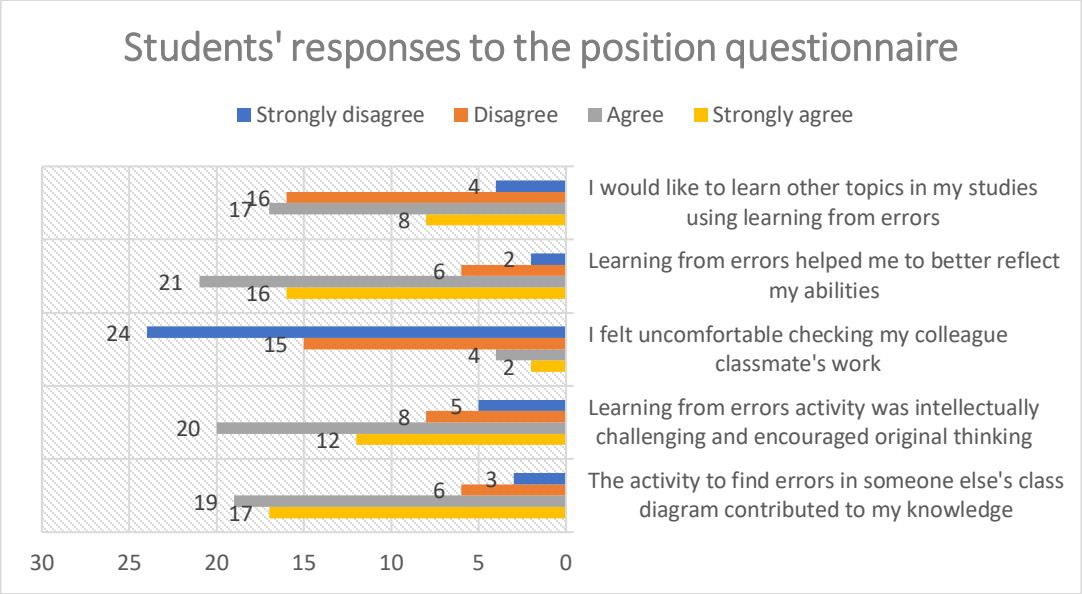


Figure 3. Student Responses to the Position Questionnaire in the Analysis and Design Course

Some quotations from the students' answer sheets concerning the learning-from-errors approach are given below:

- "Analyzing someone else's work demands a deeper understanding and higher level of controlling the course's subject."
- "As I was analyzing their work and found their errors it became clear to me what was wrong in my own work."
- "We needed to analyze another group's work, and their work wasn't clear, so it confused us about our work."
- "It is important to explain the errors to yourself. This way you get [a] better understanding of your faults."
- "You remember the errors and their cause, and the chance of repeating them is lower."

It appears that while the activity was not favorable to all students, the majority of them thought that it improved their understanding and they liked it. We now discuss the learning-from-errors approach and its implications.

5. DISCUSSION AND SUMMARY

UML diagrams are important in the process of designing a software system (e.g., Silva et al., 2017; Störrle, 2017). Class diagrams are one of the most useful types of diagrams in UML, as they clearly map out the structure of a specific system. They map the system's structure by modeling its classes, its attributes, its functionality, and the relationships between classes. The class diagrams constitute the basis of requirements specification and lay the foundation for all later design work. Therefore, their quality can have a significant impact on the quality of the system (Genero, Piattini, and Calero, 2000; Siau and Loo, 2006; Szmurło and Śmiałek, 2006; Lethbridge, 2014). Improving the quality of the class diagrams will improve the quality of software development. Correcting errors at the stage of class diagram design can help reduce development costs later in the development process (Cabot, Clarisó, and Riera, 2014). Thus, checking and searching for errors in the design process may be worthwhile. At the same time, one must consider that the design process is itself a complex, cognitive process (Villanueva et al., 2018).

The engineering curriculum makes sure to expose students to many design problems throughout their studies. This experience is meant to help students address complex problems (Marra, Palmer, and Litzinger, 2000; Dym et al., 2005). However, there are no structured programs in the curriculum for such exposure (Wankat and Oreovicz, 2015; McNeill et al., 2016; Silva et al., 2017).

Students have difficulties in designing a class diagram (e.g., Kayama et al., 2014; Bogdanova and Snoeck, 2018). They make a lot of errors and come up with inaccurate or wrong solutions. This research is based on the assumption that students can learn a lot from considering errors and that studying from errors can be useful for learning.

Our activity was conducted with novice designers and was based on students detecting errors in their colleagues' solutions in order for them to learn from those errors. Not all the students' comments to their colleagues were correct or accurate, but those comments required the students to re-examine their own solutions with a more rigorous and thorough look. The activity is based on the constructivism theory of learning that leans on the belief that knowledge construction occurs following new experiences with existing knowledge. A constructivist approach, which is built upon limited or inaccurate knowledge, may help close knowledge gaps between vague or erroneous perceptions and actual correct ones. Differences between erroneous predictions and actual outcomes raise cognitive conflicts which yield a process of reflection and critical thinking that may serve as a powerful means for reconstructing conceptual understanding. This process may transform novices' erroneous mental models into correct ones (Ben-Ari, 1998). This occurs particularly in situations where the responsibility of learning is on the learner and when the student plays an active role, as in the activity in this study.

In this activity, the students were asked to think critically about two things: (a) to decide whether the comments they received from their colleagues were correct and (b) to examine their colleagues' work and give their own comments about it. Thus, hopefully reflecting on the errors they noticed in their colleagues' work, they would apply that knowledge to their own.

Learning from errors while engaging in such learning activity may be used to correct students' erroneous conceptions regards modeling class diagrams. Previous sections presented our newly developed approach which involves an activity that provides constructive error handling, an important factor for individual learning processes (Tulis, Steuer, and Dresel, 2016). We view the learning-from-errors approach as a tool to reduce knowledge gaps.

The activity included various learning methods: working in pairs, peer assessment based on the learning-from-errors approach, discussion between the student pairs in order to reach an agreed solution, and a classroom discussion about common errors in the students' solutions. All those methods have the characteristics of active learning which has been found to be more effective in improving problem-solving skills or students' motivation than straight lecturing (Freeman et al., 2014; de Freitas, Silva, and Marsicano, 2016; Riordan, Hine, and Smith, 2017).

Working in pairs and peer assessment are important learning methods to help establish successful student outcomes and profound individual learning (e.g., Loughry, Ohland, and DeWayne Moore, 2007). In this research, the peer assessment method affects the learners in two distinguishable ways. Aside from the obvious benefit of receiving colleagues' comments, no matter how accurate they are, there is also a benefit from having to assess other work as well. Being forced to critically look at others' class diagrams helps students to develop critical thinking toward their own work.

The discussion leads students to a deeper understanding of the subject, develops their metacognitive skills, improves their ability to explain, motivates them, and influences their perceptions of success and failure. Students construct knowledge through discussions (Metcalfe, 2017). Furthermore, a discussion of errors may prevent a repetition of similar errors in the future (Borasi, 1996; Melis, 2005).

The results present the different errors made by students before and after the activity. Several important observations were made based on the error categories or the statistical analysis of the before versus after results. Table 1 shows that the number of errors in each of the categories decreased after the activity. In addition, we compared the class diagrams that the pairs designed before and after the activity. The comparison showed that the class diagrams submitted in the "After" stage were altered on the basis of the comments given to students by their peers. In light of those comments, students were successful in creating a more correct and accurate class diagram. In addition, over the two stages, students made slightly more omission errors than addition errors, but the difference appeared to be insignificant. This means that teachers should discuss "missing components," "lack of marking," and "failure to identify" in the students' class diagram designs as much as "unnecessary components" and "partial/wrong" declarations.

The statistical analysis of the results shows the relevance and potential of embedding our approach in teaching. There were significantly fewer errors in general in different aspects of the class diagram designs after the activity. After the activity, some students indicated that they realized their previous errors, of which they had been unaware. They conjectured about the origins of their errors. Quite a few students indicated that they obtained a more profound understanding as a result of the learning-from-errors activity.

Learning how to correctly design class diagrams using the learning-from-errors approach was an active and experiential learning that encouraged students to think and research. This, in turn, increased students' enjoyment and motivation. Students report that they prefer the learning-from-errors activity over the traditional teaching approach, as can be seen by students' responses to the position questionnaire. Thus, one of the benefits of our developed activities was increased student motivation.

Teachers also gain valuable information from errors. They can develop appropriate teaching methods for novices or improve their teaching methods and ways to enhance studying for novices based on these errors (Kayama et al., 2014). Teachers' tolerance for their students' errors encourages students' activity, exploration, and generative engagement (Metcalfe, 2017).

This study was conducted on a group of 45 students in the Analysis and Design of Information Systems course, a typical number for such a course. However, it would have been more beneficial if this research had been conducted with a larger number of students. While the results were found to be statistically significant, a larger sample of students should be addressed and treated with this activity in future research.

According to the typical errors found, it is possible to design assignments that will have students follow the learning-from-errors approach in order to learn class diagram design while focusing on common errors. In the future, we plan to design an assignment focused on activities to detect errors in a textual scenario presented to the students along with wrong class diagram designs. The misleading diagrams will contain deliberate incorrect solutions that stem from the errors observed or the additions and omissions detected in the mapping-errors phase. Another possibility for future work is to give the students class diagram designs with wrong statements based on observed errors and ask that they identify the errors in each statement and convince the writers of the statements that they were wrong.

In conclusion, we suggest that IS educators take a constructivist approach and develop activities such as those illustrated here in order to deepen students' knowledge while trying to reduce students' knowledge gaps.

6. REFERENCES

Alhir, S. S. (2006). *Guide to Applying the UML*. Berlin, Germany: Springer Science & Business Media.

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting Collaborative Learning and Problem-solving in a Constraint-Based CSCL Environment for UML Class Diagrams. *International Journal of Computer-Supported Collaborative Learning*, 2(2–3), 159–190.

Ben-Ari, M. (1998). Constructivism in Computer Science Education. *ACM SIGCSE Bulletin*, 30(1), 257–261.

Berardi, D., Calvanese, D., & De Giacomo, G. (2005). Reasoning on UML Class Diagrams. *Artificial Intelligence*, 168(1–2), 70–118.

Bjork, R. A. (2012). Desirable Difficulties Perspective on Learning. In *Encyclopedia of the Mind*. Edited by H. Pashler (pp. 242–244). Thousand Oaks, California: Sage.

Bock, D. B. & Yager, S. E. (2005). Using the Data Modeling Worksheet to Improve Novice Data Modeler Performance. *Journal of Information Systems Education*, 16(3), 341–350.

Bogdanova, D. & Snoeck, M. (2018). Learning from Errors: Error-based Exercises in Domain Modelling Pedagogy. In *The Practice of Enterprise Modeling: 11th IFIP WG 8.1. Working Conference, PoEM 2018*, Vienna, Austria, (pp. 321–334). New York, New York: Springer.

Booth, J. L., Lange, K. E., Koedinger, K. R., & Newton, K. J. (2013). Using Example Problems to Improve Student Learning in Algebra: Differentiating between Correct and Incorrect Examples. *Learning and Instruction*, 25(June), 24–34.

Booth, J. L., Begolli, K. N., & McCann, N. (2016). The Effect of Worked Examples on Student Learning and Error Anticipation in Algebra. Paper presented at the *38th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, Tucson, Arizona.

Borasi, R. (1994). Capitalizing on Errors as "Springboards for Inquiry:" A Teaching Experiment. *Journal for Research in Mathematics Education*, 25(2), 166–208.

Borasi, R. (1996). *Reconceiving Mathematics Instruction: A Focus on Errors*. Santa Barbara, California: Greenwood Publishing Group.

Cabot, J., Clarisó, R., & Riera, D. (2014). On the Verification of UML/OCL Class Diagrams Using Constraint Programming. *Journal of Systems and Software*, 93(July), 1–23.

Carte, T. A., Jasperson, J., & Cornelius, M. E. (2006). Integrating ERD and UML Concepts When Teaching Data Modeling. *Journal of Information Systems Education*, 17(1), 55–64.

Casterella, G. I. & Vijayasarathy, L. (2019). Query Structure and Data Model Mapping Errors in Information Retrieval Tasks. *Journal of Information Systems Education*, 30(3), 178–190.

Cherepinsky, V. (2011). Self-Reflective Grading: Getting Students to Learn from Their Mistakes. *Primus*, 21(3), 294–301.

Confrey, J. (1990). What Constructivism Implies for Teaching. *Journal for Research in Mathematics Education*, Monograph 4, Chapter 8, pp. 107–210.

Creswell, J. W. & Creswell, J. D. (2017). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Thousand Oaks, California: Sage.

Curry, L. A. (2004). The Effects of Self-explanations of Correct and Incorrect Solutions on Algebra Problem-Solving Performance. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 26, 1548.

de Freitas, S. A. A., Silva, W. C. M. P., & Marsicano, G. (2016). Using an Active Learning Environment to Increase Students' Engagement. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, Dallas, Texas, (pp. 232–236), IEEE.

Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems Analysis & Design: An Object-Oriented Approach with UML*. Hoboken, New Jersey: John Wiley & Sons.

Dranidis, D., Stamatopoulou, I., & Ntika, M. (2015). Learning and Practicing Systems Analysis and Design with StudentUML. In *BCI '15: Proceedings of the 7th Balkan Conference on Informatics*, Craiova, Romania, (pp. 1–8), New York, New York: ACM.

Dym, C. L., Agogino, A. M., Eris, O., Frey, D. D., & Leifer, L. J. (2005). Engineering Design Thinking, Teaching, and Learning. *Journal of Engineering Education*, 94(1), 103–120.

Erickson, J. & Siau, K. (2004). Theoretical and Practical Complexity of UML. *AMCIS 2004 Proceedings*, pp. 1669–1674.

Flavell, J. H. (1976). Metacognitive Aspects of Problem Solving. In *The Nature of Intelligence.* Edited by L. B. Resnick (pp. 231–235). Hillsdale, New Jersey: Lawrence Erlbaum.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active Learning Increases Student Performance in Science, Engineering, and Mathematics. *Proceedings of the National Academy of Sciences*, 111(23), 8410–8415.

Genero, M., Piattini, M., & Calero, C. (2000). Early Measures for UML Class Diagrams. *L'objet*, 6(4), 489–505.

Ginat, D. & Shmalo, R. (2013). Constructive Use of Errors in Teaching CS1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, (pp. 353–358). New York, New York: ACM Press.

Große, C. S. & Renkl, A. (2007). Finding and Fixing Errors in Worked Examples: Can This Foster Learning Outcomes? *Learning and Instruction*, 17(6), 612–634.

Gutwenger, C., Jünger, M., Klein, K., Kupke, J., Leipert, S., & Mutzel, P. (2003). A New Approach for Visualizing UML Class Diagrams. In *Proceedings of the 2003 ACM Symposium on Software Visualization*, (pp. 179–188), New York, New York: ACM Press.

Hardin, J. W. & Hilbe, J. M. (2012). *Generalized Estimating Equations*. Boca Raton, Florida: Chapman and Hall/CRC.

Henderson, C. & Harper, K. A. (2009). Quiz Corrections: Improving Learning by Encouraging Students to Reflect on Their Mistakes. *Physics Teacher*, 47(9), 581–586.

Hilbe, J. M. (2017). The Statistical Analysis of Count Data/El Análisis Estadístico de los Datos de Recuento. *Cultura y Educación*, 29(3), 409–460.

Katz, A. & Shmallo, R. (2015). Improving Relational Data Modeling through Learning from Errors. In *Proceedings of the IADIS Multi Conference of Computer Science and Information Systems MCCSIS, Theory and Practice in Modern Computing TPMC*, (pp. 198–202).

Kayama, M., Ogata, S., Masymoto, K., Hashimoto, M., & Otani, M. (2014). A Practical Conceptual Modeling Teaching Method Based on Quantitative Error Analyses for Novices Learning to Create Error-Free Simple Class Diagrams. In *2014 IIAI 3rd International Conference on Advanced Applied Informatics*, Kitakyushu, Japan, (pp. 616–622), IEEE.

Lethbridge, T. C. (2014). Teaching Modeling Using Umple: Principles for the Development of an Effective Tool. In *27th Conference on Software Engineering Education and Training (CSEE&T 2014)*, Klagenfurt, Austria, (pp. 23–28), IEEE.

Loughry, M. L., Ohland, M. W., & DeWayne Moore, D. (2007). Development of a Theory-Based Assessment of Team Member Effectiveness. *Educational and Psychological Measurement*, 67(3), 505–524.

Marra, R. M., Palmer, B., & Litzinger, T. A. (2000). The Effects of a First-Year Engineering Design Course on Student Intellectual Development as Measured by the Perry Scheme. *Journal of Engineering Education*, 89(1), 39–45.

Mathan, S. A. & Koedinger, K. R. (2005). Fostering the Intelligent Novice: Learning from Errors with Metacognitive Tutoring. *Educational Psychologist*, 40(4), 257–265.

McNeill, N. J., Douglas, E. P., Koro-Ljungberg, M., Therriault, D. J., & Krause, I. (2016). Undergraduate Students' Beliefs about Engineering Problem Solving. *Journal of Engineering Education*, 105(4), 560–584.

Melis, E. (2005). Design of Erroneous Examples for ActiveMath. In *Artificial Intelligence in Education. Supporting Learning through Intelligent and Socially Informed Technology. 12th International Conference (AIED 2005). Vol. 125.* Edited by B. Bredeweg, Ch.-K. Looi, G. McCalla, & J. Breuker (pp. 451–458). Amsterdam: IOS Press.

Melis, E., Sander, A., & Tsovaltzi, D. (2010). How to Support Meta-Cognitive Skills for Finding and Correcting Errors? In *Cognitive and Metacognitive Educational Systems: Papers from the 2010 AAAI Fall Symposium Series (FS-10-01)*, 64–68.

Metcalfe, J. (2017). Learning from Errors. *Annual Review of Psychology*, 68, 465–489.

Ohlsson, S. (1996). Learning from Performance Errors. *Psychological Review*, 103(2), 241–262.

Pinkerton, K. D. (2005). Learning from Errors. *Physics Teacher*, 43(8), 510–513.

Queralt, A. & Teniente, E. (2012). Verification and Validation of UML Conceptual Schemas with OCL Constraints. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(2), Article 13.

Ramollari, E. & Dranidis, D. (2007). StudentUML: An Educational Tool Supporting Object-Oriented Analysis and Design. In *Proceedings of the 11th Panhellenic Conference on Informatics*, Patras, Greece, (pp. 363–373).

Riordan, R. J., Hine, M. J., & Smith, T. C. (2017). An Integrated Learning Approach to Teaching an Undergraduate Information Systems Course. *Journal of Information Systems Education*, 28(1), 59–70.

Sanders, K. & Thomas, L. (2007). Checklists for Grading Object-Oriented CS1 Programs: Concepts and Misconceptions. *ACM SIGCSE Bulletin*, 39(3), 166–170.

Santagata, R. (2004). "Are You Joking or Are You Sleeping?" Cultural Beliefs and Practices in Italian and US Teachers' Mistake-Handling Strategies. *Linguistics and Education*, 15(1–2), 141–164.

Schoenfeld, A. (2009). Learning to Think Mathematically: Problem Solving, Metacognition, and Sense-Making in Mathematics. *Colección Digital Eudoxus*, 7.

Shmallo, R., Ragonis, N., & Ginat, D. (2012). Fuzzy OOP: Expanded and Reduced Term Interpretations. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, (pp. 309–314). New York, New York: ACM Press.

Siau, K. & Loo, P-P. (2006). Identifying Difficulties in Learning UML. *Journal of Information Systems Management*, 23(3), pp. 43–51.

Siegler, R. S. (2002). Microgenetic Studies of Self-explanation. In *Microdevelopment: Transition Processes in Development and Learning*. Edited by N. Granott & J. Parziale (pp. 31–58). Cambridge, UK: Cambridge University Press.

Siegler, R. S. & Chen, Z. (2008). Differentiation and Integration: Guiding Principles for Analyzing Cognitive Change. *Developmental Science*, 11(4), 433–448.

Silva, W. A. F., Steinmacher, I. F., & Conte, T. U. (2017). Is It Better to Learn from Problems or Erroneous Examples? In *2017 IEEE* 30th C*onference on Software Engineering Education and Training (CSEE&T)*, Savannah, Georgia, (pp. 222–231), IEEE.

Störrle, H. (2017). How Are Conceptual Models Used in Industrial Software Development? A Descriptive Survey. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, Karslkrona, Sweden, (pp. 160–169), New York, New York: ACM.

Szmurło, R. & Śmiałek, M. (2006). Teaching Software Modeling in a Simulated Project Environment. In *9th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2006)*, Genova, Italy, (pp. 301–310), New York, New York: Springer.

Tulis, M., Steuer, G., & Dresel, M. (2016). Learning from Errors: A Model of Individual Processes. *Frontline Learning Research*, 4(2), 12–26.

Villanueva, I., Campbell, B. D., Raikes, A. C., Jones, S. H., & Putney, L. G. (2018). A Multimodal Exploration of Engineering Students' Emotions and Electrodermal Activity in Design Activities. *Journal of Engineering Education*, 107(3), 414–441.

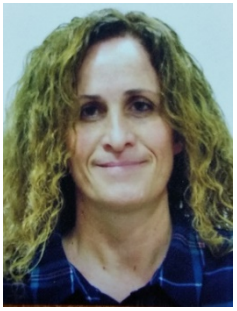
Wankat, P. C. & Oreovicz, F. S. (2015). *Teaching Engineering. (2nd ed.)*. West Lafayette, Indiana: Purdue University Press.

Watson, R. T. (2006). The Essential Skills of Data Modeling. *Journal of Information Systems Education,* 17(1), 39–42.

Yerushalmi, E. & Polingher, C. (2006). Guiding Students to Learn from Mistakes. *Physics Education*, 41(6), 532–538.

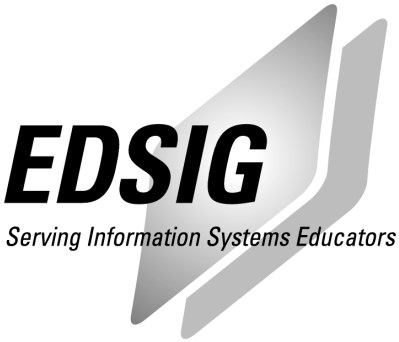
AUTHOR BIOGRAPHIES

Ronit Shmallo is a lecturer and researcher in the department of industrial engineering and management at the Shamoon College of Engineering. She received her Ph.D. in computer science education from Tel-Aviv University in 2013. She teaches primarily computer science programming and the analysis and design of information systems using an object-oriented approach. Her research focuses on difficulties encountered by novices in trying to understand the cornerstones of object-oriented programming. Her study involves an examination of a new teaching method that integrates explicit orientation to errors in a way that enables students to learn from those errors in different topics.



Tammar Shrot is a lecturer and researcher in the department of software engineering at the Shamoon College of Engineering. She received her Ph.D. in computer science from Bar-Ilan University in 2013. She teaches primarily computer science programming using an advanced object-oriented approach, the analysis and design of software systems, and artificial intelligence. Her research focuses on computer–human interactions, intelligent user interfaces, and the complexity of manipulating tournaments and voting. Her study involves an examination of learning algorithms in the concepts of user interfaces and the examination of the complexity of manipulation protocols.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 2574-3872