Learning from Errors: Error-based Exercises in Domain Modelling Pedagogy

Daria Bogdanova, Monique Snoeck

Learning from errors: error-based exercises in domain modelling pedagogy

Daria Bogdanova[1] and Monique Snoeck[1]

[1]Research Center for Management Informatics, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
daria.bogdanova@kuleuven.be; monique.snoeck@kuleuven.be

Abstract. Conceptual modelling remains a challenging topic for educators, as it concerns ill-defined problems and requires substantial amount of practice for reaching even the initial level of proficiency. Year after year, novice modellers tend to make similar errors when learning to design models and some of those errors become persistent even at the higher level of proficiency. Are these errors the unavoidable "necessary evil" or there is a possibility to address them at the very early stage of a modeller's education? In this work, we examine a novel approach to teaching conceptual modelling by identifying the most frequent errors in students' models and introducing error-based step-by-step exercises in the framework of a Small Private Online Course for university students.

Keywords: Conceptual Modelling, Domain Modelling, Enterprise Modelling, Education, Error-Based Learning, Adaptive Expertise, UML, Class Diagrams

1 Introduction

The question of properly addressing students' errors in the subjects rich with ill-defined problems is one of the substantial challenges arising before educators. In conceptual modelling pedagogy, this question is of a particular significance, as the novice modelers should not only reach the "routine"-level expertise that implies knowledge of a repertoire of tools or procedures, but also become adaptive experts that are able to promptly grasp the core of provided requirements, identify the changes in the previously learned task, and adapt the procedures accordingly.

Although numerous guidelines, reusable patterns and other materials on conceptual modelling (and, specifically, on UML class diagrams) are available, novice modelers tend to struggle with grasping the gist of the subject. Moreover, the extensive amount of materials may even hinder the development of a novice – "typical novice analysts fail to derive maximum benefit from such assistance due to the cognitive overload involved in the recommendations and guidelines"[1:108].

In addition to the challenge of the "cognitive overload", novice modellers are often provided with unbalanced learning material, which is focused either on the lowest-level cognitive skills (e.g. "understand" level, according to the revised Bloom's taxonomy [2]), or the highest, such as the very creation of a model "from scratch", while the intermediate levels necessary for a constructive skill acquisition involving learning to

apply procedures, analyse and evaluate models and their parts, remain underrepresented in the pedagogical materials [3].

The abovementioned difficulties require thorough reflection and action at least at a level of a particular university course, and ask for rethinking of conceptual modelling curriculum fieldwide.

In this paper, we will take a closer look particularly at domain modelling errors that students tend to make, propose an error-based approach to creating step-by-step modelling exercises and evaluate the preliminary result of its implementation in the context of a master-level course of Architecture and Modelling of Management Information Systems at KU Leuven. We will examine the effectiveness of targeted step-by-step online exercises for preventing the most common student errors in simple UML models design at a task level and identify the content areas and concepts, which cause most difficulties.

2 Background

2.1 Knowledge evaluation criteria

The quality of a model designed by a student can be considered the most important indication of mastery of the subject. One of the most commonly accepted modelling quality frameworks is the three dimensional framework proposed by Lindland et al. [4]. The framework proposes to evaluate a conceptual model from three quality perspectives: syntactic (formal syntax of the model), semantic (relevance of the model to the domain it describes) and pragmatic (readability/understandability of the model). Thus, errors in modelling can be classified according to the quality dimensions they belong to, both in the professional and educational settings. However, in an educational setting dedicated to the initial stages of training and design of simple models, more narrow evaluation criteria can be applied, so that students could reflect not only on the final modelling solution, but also on the flaws in the various stages of modelling, and/or be informed on the specific content area that requires revising. As an example, in [5], a simplified set of criteria suitable for novice learners of simple class diagrams is proposed, including the syntactic, class-related, attribute-related and association-related errors. If classified according to [4], these types are part of only syntactic and semantic quality dimensions, with no pragmatic dimension involved. However, those two dimensions of quality are considered the most important at the initial stage of learning, when the students must grasp the core principles of modelling. Afterwards, students should be able to refine the semantically and syntactically valid model according to the pragmatic quality standards.

2.2 Novices' errors in domain modelling

Identification of typical modelling errors has been subject of a number of studies in the last two decades. Novice modelers tend to struggle with similar types of tasks and notions throughout time. In 1994, an experimental study on novice errors in conceptual

database design showed that the typical errors included literal translation of requirements, bias related to incomplete knowledge, errors in relationship degree, as well as incorrect connectivity (“one” or “many”) [6]. Similar errors were found in 2005 by Leung and Bolloju, who performed a detailed analysis of the quality of domain models developed by novice systems analysts [7] based on the Lindland et al. [4] model quality framework and studied the interrelations between the pairs of commonly occurring errors. According to their findings, the most common errors were related to semantic and pragmatic quality, with the most popular error in the category “unexpected is presented", which means that the novice modelers tend to overload the model with unnecessary entities or attributes. The most frequent semantic error was placing the wrong cardinality or multiplicity. The syntactic errors were also quite common (about the quarter of the errors, overall), despite the fact that an automated tool was checking the model syntax for the students prior to submission.

Although the solutions and recommendations proposed by researchers and educators regarding common and recurring errors differ in detail, there is a consensus on the very need for modification of modelling pedagogy regarding those errors, as every paper found had a suggestion regarding such modification. Several successful attempts to employ teaching methods based on common modeling errors have been reported. For instance, a quantitative error analysis of class diagrams created by university freshmen and subsequent modification of the teaching method with greater focus on most common errors (syntactic, attribute-related, association-related and class-related), led to the “improved performance related to syntactic errors and relation errors in fundamental tasks” [7:621]. An analogical use of a “prophylactic approach” to teaching UML provided improved results in summative quizzes developed to test the competences of students in modelling relationships between classes, requirements identification and creating a simple class diagram [9].

2.3 Technology-enhanced learning support

Another approach to dealing with novice errors is providing immediate feedback on the simple models designed by students. The ability to create simple class diagrams (by “simple” we imply those consisting of up to five classes) without errors can be considered a fundamental first step for mastering conceptual modelling. On the level of a simple model, where the variety of possible valid solutions is still much more limited and the model solution is easily available, the use of intelligent tutoring systems (ITS) or other educational software becomes possible. In [10], an implementation of a sample solution-based ITS for teaching UML skills, with a pre-built set of possible error messages, resulted in no worse result than a traditional learning setting, while providing a more enjoyable experience for students and reducing teacher’s time on correcting students’ solutions. In [11], the use of technology-enhanced support with implementation of immediate automated feedback in a conceptual modelling course resulted in improvement of students’ performance, as well as the positive student perception of the course.

In modelling pedagogy, technology is employed not only at the task level, but also throughout the whole modelling curriculum – for example, by means of MOOCs (Massive Open Online Courses) or SPOCs (Small Private Online Courses). MOOCs on conceptual data modelling remain not numerous, with just a few available for the wide audience [3]. Typically, such courses consist of a number of modules that include videos with theoretical and practical materials and practice exercises at the end of each module – in a form of a multiple-choice quiz or other formative or summative task with automated assessment. Such a variety in types of materials and sequencing of tasks and theory lessons provides additional educational opportunities both for the students and for the educators and could be leveraged to provide error-based support. However, none of the currently available online courses on modelling provides students with gradual step-by-step exercises specifically on conceptual modelling, and UML diagram design, in particular.

3 Methodology

3.1 General Approach

The course Architecture and Modelling of Management Information Systems is taught to the master students of the faculty of Business and Economics at KU Leuven. The course has been successfully taught for over a decade, evaluated and improved after each iteration. A thorough evaluation of student mistakes was made in 2017 to propose a targeted improvement the next year. The targeted improvement was performed in 2018 by introduction of step-by-step error-based formative exercises in an online course. The improvement was set up according to an experimental design, such as to be able to evaluate the effectiveness of the proposed improvement by comparing students' performance in 2018 to the performance of the 2017 cohort. In particular, care was taken to isolate the treatment and keep the rest of the course similar to the 2017 setting as much as possible.

3.2 Subjects and general setting

Two similar groups of master students (39 students in 2017 and 32 in 2018) from the same trajectories and following the same set of mandatory courses followed the course of Architecture and Modelling of Management Information Systems. The course includes an extensive module on UML class diagram design following the MERODE approach [12] and employs the JMermaid modelling software that provides students with immediate automated feedback. As part of the course, students are required to complete a series of exercise sessions and submit the solution of provided cases.

Student groups are very similar across the successive academic years, in particular concerning variables that might influence their modelling skills. In terms of language skills, the course is taught in English, which is a second language for the very large majority of the students. To be accepted to the master program and, subsequently, to

the particular course, the international students have to pass a unified English proficiency exam, thus, we assume that the students in both groups possess sufficient mastery of English language to understand the tasks and the requirements provided in the course equally. In terms of prior education on modelling or other IT skills, all students have very limited experience in these matters as the master program is intended for academic bachelors with a non-IT background.

The course materials on theory were identical for both groups of students, however, in 2018 a Small Private Online Course was introduced to provide formative exercises and ensure better understanding of the subject.

3.3 Instructional design

Throughout the course, 4C/ID instructional design model was applied. 4C/ID is a model developed specifically to design training programs aimed at complex skills [13]. The key parts of the model are: a sequence of learning tasks (whole-task practice – authentic learning experience), supportive information, just-in-time information (including examples and corrective feedback) and part-task practice (practice for a selected skill with tasks of a narrower focus). In the Architecture and Modelling of Management Information System course, the learning tasks (whole-tasks) are represented by complete cases, where students have to build a model based on textual requirements. Supportive information is provided in the textbook, presentations or in the online course: the information is doubled throughout different resources, so the students could choose the most convenient one. Just-in-time information is provided by means of automated feedback in the modelling software and/or by means of automated feedback in the SPOC exercises, while part-task practice is provided during the exercise sessions either by means of the modelling software during a collective exercise or in the SPOC.

In 2017, after the presentation of theoretical material and examples, the students solved two cases during a lab sessions. The two exercises had an identical set-up: in the course of the session, the students were given automated feedback of two types by the modelling software – a reminder to simulate the model after certain amount of actions, and a multiple choice question provoking the reflection on an association just created by the student.

In 2018, the "treatment" constituted of using the identical cases as in 2017, but providing part-task practice for the first case by means of step-by-step online exercises in a SPOC. The second case was (similarly as in 2017) given “as a whole”, without subdividing it into part-tasks. This allows to assess to what extent the students were able to extrapolate the practical experience received in the first case to the second one. The effectiveness of the treatment can then be assessed by measuring the improvement on the second exercise in 2018, compared to the 2017 performance.

3.4 The cases

The students were asked to solve two cases provided requirements documents. The cases were designed to test the ability of students to understand and apply the following concepts and elements:

- Correct identification of classes and associations from the requirements document
- Inheritance and the notion of roles
- Correct multiplicities of associations

Each of the model solutions included: five or six classes, with one central element connected with a chain of two or three classes, a single class and a class with a recursive association. The multiplicities in the two cases differed according to the specific requirements given in the task.

The model solutions of the two cases are provided in Fig. 1 and Fig. 2.

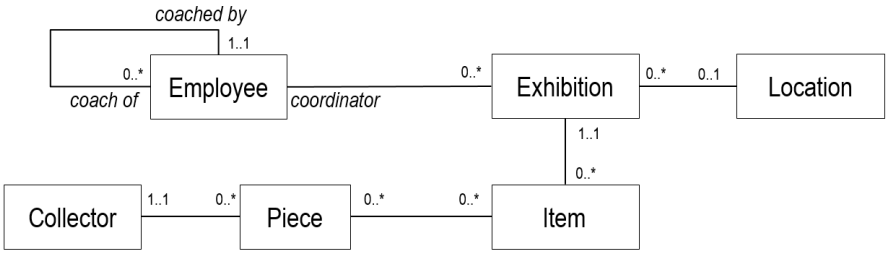

Fig. 1. Model solution for Case 1, central class being Exhibition

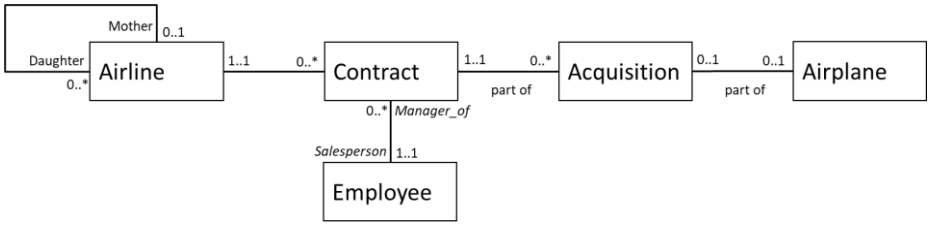

Fig. 2. Model solution for Case 2, central class being Contract.

The full description of the cases can be found in Appendix A

3.5 Frequent errors identification

The student solutions of the exercise session cases became the source for a frequent errors collection. The error identification resulted in the following error types related to classes and associations (the corresponding quality dimension according to the Lindland et al. [4] is mentioned in parentheses):

Class-level errors:

1. No meaningful name is given to a class (Pragmatic)
2. Missing classes (Semantic)
3. Superfluous classes (Semantic)

Association-level errors:

4. No meaningful name is given to an association (Pragmatic)
5. Missing association (Semantic)
6. Superfluous association (Semantic) – see Fig. 3
7. Name-concept mismatch (Semantic)
8. Wrong multiplicity (Semantic)
9. Unnecessary reification (Semantic)
10. Role inversion/Degree (Semantic)
11. Wrongly linked association (Semantic)

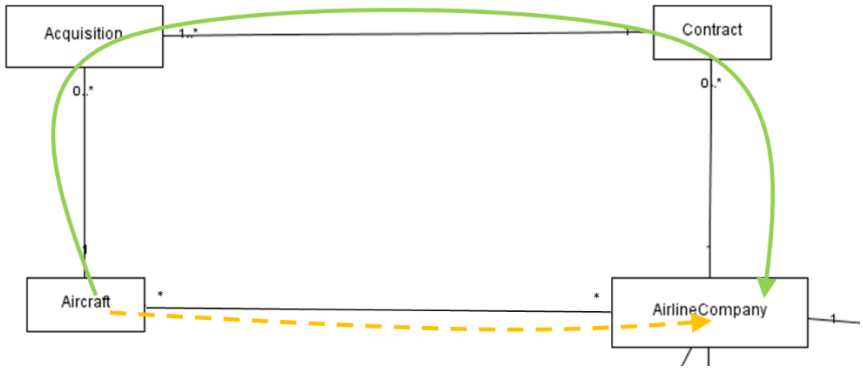

Fig. 3. An example of a superfluous association (dashed arrow) from a student solution: the airline company that owns the aircraft (yellow dashed path) is the airline company that placed the contract for the acquisition of the aircraft (green path).

When modelling associations, students are requested to think about the relationships between the life cycles of the objects, and in particular to reflect about which objects need to exist first, whether associations ends are frozen or not, and what objects need to be deleted first. For example, when modelling the association between AirlineCompany and Contract, the student should realize that before a Contract can be registered, there needs to be a AirlineCompany (or the AirlineCompany needs to be registered simultaneously), that the contract cannot "change" AirlineCompany throughout its life and that it cannot exist any longer than the AirlineCompany-object it refers to (meaning that the association end is frozen). In ER-terms, the weak entity (Contract) will need the strong entity (AirlineCompany) to exist first, and cannot outlive it.

Some clarification is necessary for error types 7, 9, 10 and 11:

"*Name-concept mismatch*" refers to the problems where an association has been reified to an association class, and the name of the association class does not convey the meaning of the association. A typical example is the unary "partnership" association between airlines. If reified to an association class, its name should reflect the fact that the class represents a partnership. If the association class is named e.g. "daughter_airline", this represents a name-concept mismatch.

"*Reified too often*" refers to an association that has been reified to an association class, and whereby one of the resulting new associations has been reified again. A typical example is the association between employee and contract, giving rise to an association class "SalesManagerDuty" (with attributes such as start date, end date, etc.). If then an association between "SalesManagerDuty" and "Contract" is reified again, such reification is excessive. Weak associations that express existence dependency should not be reified.

"*Inverted roles*" refers to the fact that the student made a wrong analysis, and indicated the wrong class as the "strong" vs "weak" entity in the association. For recursive associations, a "degree" problem refers to the fact that the association was not modelled as a recursive association, but rather an extra class was created to which the class airline was linked. Both types of problems refer to the fact that a student does not manage to make a correct in-depth analysis of the semantics of the association s/he is drawing.

Finally, "*wrongly linked*" associations refer to classes linked wrongly, such as linking Airplane to Airline rather than to Contract. These errors also result from missing classes (e.g. the Airplane directly linked to Contract because of the missing Acquisition class).

As it can be seen from the list, most of the errors are related to the semantic quality dimension. The small amount of syntactic errors is explained by the fact that the JMermaid tool prevents the input of models with syntactical errors.

The student solutions of Case 2 from both academic years (2017 and 2018) were checked and marked according to the list of errors identified in 2017.

3.6 Step-by-step online exercises

A set of step-by-step online exercises was designed for the iteration of the course in 2018 to prevent most of the commonly occurring errors in class diagram design, as identified in the previous academic year. The steps provided in the online learning platform as case-related guided exercises, had to be reproduced by the students in the second case afterwards, without guidance.

The exercises included the following:

1. Identifying enterprise object types – students could choose several options that they believed were object types according to the requirements document. This exercise aimed to address error types 1, 2 and 3.
2. Modelling associations – multiple-choice test based on given case requirements to address error types 5, 6 and 11.
3. "People and their roles" – and multiple choice exercise aimed at differentiation between a base concept and a role, to address error types 3 and 10.

No specific treatment was given for error type 4, as the semantic quality of the models was given a higher priority. Errors 7 and 9 (name-concept mismatch and reification problems) were not addressed in the online course due to the complex nature of the problems related to those errors, which is hard to address in an entirely automated way. Error 8 (multiplicity error) was not treated specifically for these tasks, as the theoretical material as well as a number of exercises related to the topic of multiplicity were presented to the students previously in the course.

Immediate automated feedback on the answers was provided to the students. An example of and exercise with feedback can be seen in Fig. 4.

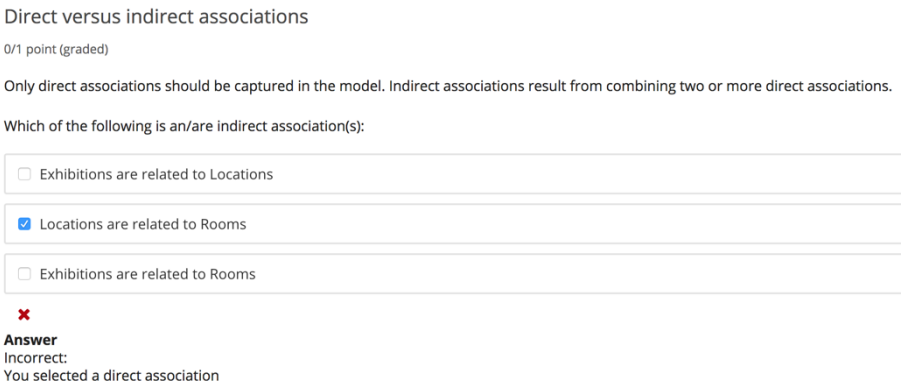

Fig. 4. Part of a guided exercise with feedback.

4 Results

This part presents the summary of student solutions analysis for Case 2 and the comparison of the solution quality to the model solution (see Fig. 2).

4.1 Class-level errors

Table 1. Summary of the class-level errors

Class-Level	2017								2018							
	Contract	Airline	Employee	Acquisition	Airplane	General	Total	Task Frequency	Contract	Airline	Employee	Acquisition	Airplane	General	Total	Task Frequency
Pragmatic Quality																
Problem with name			0,36	0,08			17	44%		0,06	0,66	0,19			29	78%
Semantic Quality																
Missing class		0,10		0,31	0,18		23	59%				0,13			4	13%
Superfluous class(es)						0,31	12	31%							0	0%
Relative frequency/totals	0,00	0,10	0,33	0,38	0,18	0,31	51	1,308	0,00	0,06	0,66	0,31	0,00	0,00	*33*	*1,03*

Table 1 gives an overview of the class-level errors found in student solutions. The total count of error occurrence is listed in the "Total" column for each error type. The "Task frequency" column indicates the percentage of tasks where the errors of certain type occurred (one or several times). The columns with class names indicate the relative frequency of an error: the number of times the error occurred divided by the total number of tasks (student solutions).

Name problems. Whereas in 2017 there is an average of 44% of the tasks showing name problems, the frequency in the 2018 solutions is much higher. This increased frequency is mostly due to students using the role name "SalesPerson" as a name for the class "Employee", and subsequently using "Assignment" or "Management" as role names.

Missing classes. In the 2017 solutions, this error occurs 23 times, and in 59,0% of the tasks, whereas in 2018, this error appears only 4 times and in 13% of the tasks.

Superfluous classes. In 2017, there are 12 occurrences of the error, appearing in 30.8% of the tasks. In 2018, there are no superfluous classes in the proposed solutions. On average, there are 1.31 class-level errors per task in 2017, with approximately 2 out of the 3 errors being semantic quality problems. In 2018, we see on average 1,03 error per task, the large majority of which (29 of 33) are naming errors (pragmatic quality).

4.2 Association-level errors

Overall, one can immediately see from Table 2 that association-level problems have a much higher frequency than class-level problems.

Missing associations. Obviously, when a class is missing, any association that would involve this class is missing as well. Therefore, we only counted the additional missing associations when the required class(es) were present, but the association was missing. In 2017, in three solutions the recursive association representing airline partnership was missing. In 2018, this happened only once.

Superfluous associations were always the result of superfluous classes, so these errors were not counted separately. There were no solutions where an additional association was added on top of the required associations between two correctly captured classes.

Table 2. Summary of the association-level errors

Association-Level	2017								2018							
	Contract-Airline	Airline-Unary	Contract Employee	Contract-Acquisition	Acquisition-Airplane	General	Total	Task Frequency	Contract-Airline	Airline-Unary	Contract Employee	Contract-Acquisition	Acquisition-Airplane	General	Total	Task Frequency
Pragmatic Quality																
Problem with name		0,05	0,21				10	26%		0,09	0,31				13	41%
Semantic Quality																
Missing association		0,08					3	8%		0,03					1	3%
Superfluous association							0	-								
Name-concept mismatch		0,13	0,03				6	15%		0,34					11	34%
Multiplicity problem	0,13	0,18	0,26	0,03	0,56		45	95%	0,06		0,09	0,06	0,81		33	94%
Reified too often	0,03		0,08	0,05			6	15%	0,03						1	3%
Role inversion/Degree	0,08	0,21	0,31	0,03	0,05		26	54%	0,03	0,06	0,06	0,06	0,06		9	28%
Wrongly linked	0,13	0,08	0,08		0,38		26	51%	0,06		0,06	0,03	0,19		11	22%
Relative frequency/totals	0,31	0,59	0,69	0,10	0,79		*122*	*3,13*	0,15	0,41	0,33	0,08	0,82		*79*	*2,469*

Name-concept mismatch. In 2017, we find this problem occurring mainly for the recursive association on Airlines, and three times for the Contract-Employee association. In 2018, the relative frequency is higher, especially for the recursive association on Airlines.

Multiplicity seems to be the most complicated concept to get right, as more than 90% of the tasks suffer from this problem in both years.

Reified too often error appears 6 times and in 15,4% distinct tasks in 2017, and only once in 2018.

Inverted role/degree problems occurred 26 times in more than a half (54%) of the distinct tasks, as opposed to only 9 times in 28% distinct tasks in 2018.

Wrongly linked associations occurred 26 times in more than half (51%) of the distinct tasks, as opposed to only 11 times in 22% distinct tasks in 2018.

In total, the overall amount of association-level errors decreased – from 122 errors in total (3,13 errors per task) in 2017 to 79 (2,47 errors per task) in 2018.

5 Discussion

There are a number of limitations that should be considered regarding this study. First of all, it should be viewed as a small-scale exploratory analysis, as the groups of students were relatively small (~30 to 40 persons in each group). However, such group size is typical for university exercise session setting. Nevertheless, larger population of students should be addressed and treated with step-by-step exercises in the future research, e.g. by means of a MOOC. Second, the focus of this study is narrowed to a specific type of modelling tasks, in order to capture the granular view of the learning process. In the future, it may be beneficial to "zoom out" to the level of the entire course and check the impact of step-by-step exercises for modelling on student performance throughout the course. Third, the error detection was done for the exercise that followed the previous one immediately (Case 2 was given in the same exercise session as Case 1 for both years). Thus, the long-term effectiveness of the step-by-step exercises is yet to be determined. Also, more focus could be given to improving the pragmatic quality of student models, as in the current version of the course, semantic quality was considered the key problem to tackle, with no specific exercises for improving the pragmatic quality.

As it can be seen from the preliminary results, the group that was treated by step-by-step exercises showed improved results in comparison with the group of the previous years in all types of errors, except for pragmatic (name-concept mismatch).

Class-level errors are less numerous than association-level errors in both years, which is consistent with the findings of previous studies [1, 8]. The name problems were more common in 2018 than in 2017, though compensated by much less semantic errors. The increased number of wrongly named classes in 2018 is mostly located at the level of the class employee. This may have been induced by the part-task training on roles, as many students named the class "Employee" as "SalesManager" instead. This calls for a revision of the corresponding exercise to better emphasize the need for a

correct name for the base class. The number of missing classes in 2018 dropped significantly in comparison with 2017, while the superfluous classes error was completely eliminated in 2018, which might be the result of the targeted exercise in the online course that provided students with an opportunity to reflect on the choice of classes from the textual description. In average, there are less class-level errors in 2018 than in 2017.

On association level, the group of 2018 also outperforms the group of 2017. The most common error in the association level was wrong choice of multiplicity, which in both years occurred in the vast majority of the tasks. Multiplicity errors are common in various types of modelling exercises reported in other sources [6, 7]. Missing recursive associations, as well as unnecessary reification, were several times less common in 2018 than in 2017, which can suggest that the targeted exercise showed its effectiveness in training the association-level skills. However, since the missing associations were, obviously, not counted for the classes that were missing, this implies an underrepresentation of association errors for 2017 compared to 2018, as in 2018 there were a few missing classes, while there were much more missing classes in 2017.

Overall, concerning task-specific errors, from the totals in each column of Table 2, it is easy to see that the recursive association on “airplane”, the “acquisition”-to-“contract” and the "sales representative" association between employee and contract are the most difficult ones to capture correctly. This calls for an additional part-task training on recursive associations. At the same time, the error frequencies also demonstrate that the "part-task" exercise, in which the students were requested to analyze in detail three potential associations, can be viewed as a way of substantial improvement for the course. Errors that indicate shortcomings in understanding the semantics of an association and the roles classes play in the association as witnessed by name-concept mismatches, inverted roles, degree problems and wrongly linked associations, are much less frequent in 2018 than in 2017.

6 Conclusion and Future Research

In this work, we have implemented a series of step-by-step exercises based on known common errors to teach a specific part of the modelling course – building a simple UML model with a recursive association and a chain of associations based on textual case description. Summing the results presented above, we can make a preliminary conclusion that the step-by-step exercises implemented in a Small Private Online Course in the framework of 4C/ID instructional design model have shown to be effective, at least when it concerns the immediately following exercise. Nevertheless, while the majority of errors (semantic) both on class and association levels seem to be tackled successfully in the latter group of students, there were two error types – name-concept mismatch (pragmatic) and multiplicity errors – that require thorough investigation and targeted training.

We are planning to continue the implementation of 4C/ID model and introduction of more step-by-step online exercises in the course, and to check whether the improvement on the task level will extrapolate to the course level, as well. In addition, the current

SPOC on enterprise information systems modelling will be expanded into a MOOC to address larger groups of students, so it will be possible to investigate and improve this type of pedagogical approach further, and make a larger and more representative collection of student errors.

7 References

1. Bolloju, N., Leung, F.S.K.: Assisting novice analysts in developing quality conceptual models with UML. Commun. ACM. 49, 108–112 (2006).
2. Krathwohl, D.R.: A Revision of Bloom's taxonomy. Theory Into Pract. (2002).
3. Bogdanova, D., Snoeck, M.: Domain modelling in bloom: Deciphering how we teach It. (2017).
4. Lindland, O.I., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. IEEE Softw. 11, 42–49 (1994).
5. Kayama, M., Ogata, S., Asano, D.K., Hashimoto, M.: Educational Criteria for Evaluating Simple Class Diagrams Made by Novices for Conceptual Modeling. Int. Assoc. Dev. Inf. Soc. (2016).
6. Batra, D., Antony, S.R.: Novice errors in conceptual database design. Eur. J. Inf. Syst. 3, 57–69 (1994).
7. Leung, F., Bolloju, N.: Analyzing the Quality of Domain Models Developed by Novice Systems Analysts. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences. p. 188b–188b. IEEE.
8. Kayama, M., Ogata, S., Masymoto, K., Hashimoto, M., Otani, M.: 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. pp. 616–622. IEEE (2014).
9. Elva, R., Workman, D.: A Prophylactic Approach to Teaching UML in Undergraduate Computer Science Courses. In: The Fifteenth International Conference on Learning (2008).
10. Schramm, J., Strickroth, S., Le, N.-T., Pinkwart, N.: Teaching UML Skills to Novice Programmers Using a Sample Solution Based Intelligent Tutoring System. Flairs '12. (2012).
11. Sedrakyan, G., Snoeck, M.: Technology-enhanced support for learning conceptual modeling. In: Lecture Notes in Business Information Processing. pp. 435–449. Springer, Berlin, Heidelberg (2012).
12. Snoeck, M.: Enterprise Information Systems Engineering: The MERODE Approach. Springer Publishing Company, Incorporated (2014).
13. Merriënboer, J.J.G., Jelsma, O., Paas, F.G.W.C.: Training for reflective expertise: A four-component instructional design model for complex cognitive skills, (1992).

Appendix A. Cases descriptions

Case 1. Mouvre Museum

The Mouvre Museum in Paris is a huge museum with quite a large number of rooms, so that many exhibitions can be organised in parallel in the Museum. Also, the planning phase of an exhibition starts at least two years before the actual opening date of an exhibition, so that even for a single room, several exhibitions in different stages of advancement need to be followed up simultaneously. Therefore, a little management system is required to make sure all these exhibitions run smoothly.

The museum has identified a set of locations inside the museum that can hold exhibitions. The locations can be considered as museums inside the museum. So, for each location a series of exhibitions is developed. For each exhibition, first a series of desired exhibition items is defined. For example, for an exhibition on Vincent Van Gogh, it is defined that one item of his early period is desired, one pencil drawing with the corresponding painting, one sunflower painting, etc. For each desired item, a suitable piece is sourced from the collectors that possess candidate pieces. For some items, only one unique piece is available, but some exhibition items several potential pieces are available from different collectors. (There are for example several "Sunflower" paintings from Vincent Van Gogh). For each exhibition item, the system will keep track of what pieces are requested from which collector.

Each exhibition is assigned an employee of the museum as coordinator. To foster knowledge transfer, junior employees are assigned a senior employee as coach.

Case 2. Boncardier

Boncardier sells aircrafts to airline companies. As aircrafts are very expensive to build, they are only built "on demand", meaning that first a sales agreement is made with a customer, before the airplane is actually built. (An exception are demo versions of airplanes, but these are out of scope for this case). The sales are regulated by means of contracts with the airline companies, whereby a single contract may consist of several acquisitions of airplanes. The global contract stipulates common elements across all acquisitions such as delivery conditions, legal aspects, etc. Each acquisition of an airplane has further specific details, such as the chosen model of airplane, the negotiated price for that airplane, chosen options & customizations, delivery date, etc. Each contract is managed by a Boncardier salesperson. An employee can act as salesperson for several contracts. Given the long term of contracts, the assigned salesperson may change over time, but Boncardier ensures there is always a salesperson available for the client.

Some airlines are related to each other: for example, main airlines often have a low cost daughter airline company. Boncardier therefore keep track as much as possible of the mother-daughter relationships between airline companies, to be able to track whether to sold aircrafts are shifted to partner airlines of the original buyer.