Teaching Sequence Diagrams To Programming Beginners And The Change Of Algorithmic Conceptions

Philipp Shah
Technische Universität München
TUM School of Education
Munich, Germany
Philipp.Shah@tum.de

Peter Hubwieser
Technische Universität München
TUM School of Education
Munich, Germany
Peter.Hubwieser@tum.de

Abstract— Computer science students could have different concepts and misconceptions about the order of statements of object-oriented programming code and their effects. This paper shows the positive influence of early teaching UML sequence diagrams to programming beginners to their concepts of programming statements and the order thereof.

Keywords—UML sequence diagrams; misconceptions; computer science concepts; statements; order; qualitative content analysis.

I. INTRODUCTION

Students of computer science have different conceptions of the order of statements and their effects. Misconceptions like ‘the order of statements does not change the runtime’ or ‘after a declaration, it is sufficient that the initialization of a reference attribute follows an execution of a method of this object’ could affect a programming beginner. This small qualitative study shows how the conceptions of statements and their order could change after UML sequence diagrams are introduced right at the beginning of object-oriented lessons. It shows that sequence diagrams have a positive influence on programming beginners and their conception of statement order. At least for the small group of 60 students of two different classes, who were the basis for this paper, misconceptions disappeared after sequence diagrams were introduced and practiced. That implies that teaching UML is more than teaching modeling techniques and diagrams. It also has positive effects on programming skills.

II. RELATED WORK

A. Misconceptions in General

According to Stavy and Tirosh, a misconception is a conception that is not in line with accepted scientific notions [1], e.g. some students are stating that the manipulation of a counter variable of a loop after the loop had been done, still affects the loop [2]. Or “if you have two different variables, they must refer to two different objects”. Holland, Griffiths and Woodman called this category “Identity/attribute conflation” [3].

It is important to mention that not every single programming error has its origin in a misconception of the programmer. If a student just forgets a semicolon at the end of a simple assignment, the student might not have a misconception at all.

B. Order of Statements

Some misconceptions address the order of statements and the effects thereof like “two programs containing the same statements (even if in different order) are equally efficient”, discovered by Gal-Ezer and Zur in 2004 [4] and verified by Shah, Capovilla and Hubwieser in 2015 [2]. In addition, some students state that even after a loop is completed the manipulation of the limits of the counter variable still affects the loop [2].

Problems with order of statements and related misconception are not new to computer science education. 2005 Ragonis and Ben-Ari discovered that some high school students might think that “Methods are executed according to their order in class definition” [5]. They analyzed high school students’ answers and programs while interacting with them [6] using a constructivist qualitative research methodology.

C. Sequence Diagrams

As sequence diagrams are visualizations of the order of method invocations and their response, we analyzed their influence on the conceptions and misconceptions which are corresponding to statement order. It has already been published a small variety of literature about sequence diagrams in computer science education.

Xie, Kraemer and Stirewalt in 2007 analyzed the influence of teaching UML sequence diagrams to support the understanding of thread interactions and concurrency concepts. Therefore, they adjusted the standard UML notation of sequence diagrams for their needs and tested the implication of the new sequence diagrams on undergraduate university students [7]. They were able to show a positive influence of teaching sequence diagrams to solving concurrency problems. Later, in 2009, Xie, Kraemer, Stirewalt, Dillon and Fleming proved that students who had been taught the adjusted sequence diagrams

© IEEE, 2017. This is the author's version of the work. It is posted here by permission of IEEE for your personal use. Not for redistribution. The definitive version was published in EDUCON, (2017)
https://doi.org/10.1109/EDUCON.2017.7942863
D. UML Diagrams in General and other Diagrams

Sequence diagrams are not the only kind of diagrams which could help programming beginners to learn object-oriented programming. Therefore researchers also analyzed other diagrams and their benefit for teaching computer science.

Al-Fedaghi and Alrashed referred to Barros, Biscaia and Vitória [9] and introduced a new diagram different from known UML diagrams [10] in 2014. Unfortunately, this new kind of diagram is much more complex than sequence diagrams and focused on explaining students how a high-level command is executed on a hardware level.

In 2005 Engels, Hausmann, Lohmann and Sauer [11] presented a course design for undergraduate CS students, which showed the use of UML diagrams for teaching core concepts of software engineering. They conclude that UML is more than “just another modeling notation”. According to them it is an “excellent vehicle” to illustrate core concepts of computer science and software engineering.

Also in 2005 Turner, Pérez-Quiñones and Edwards [12] introduced a program (called minimUML) that provides just a minimum set of UML, which could be used for programming beginners. The goal was to support novice students creating UML class diagrams in an easy way. Additionally the student is able to use a code generation function, to generate C++ or java code out of the designed class diagram. An evaluation of the usefulness of this program was done with a small group of students.

III. STUDY

We analyzed the influence of teaching UML sequence diagrams very early to the students’ concepts of object-oriented programming, especially on concepts in relation with statement order.

We ran our survey six weeks after the start of the new school year. All students had computer science the year before, in which they learned functional modeling, computation with spreadsheets, database modeling and writing queries with SQL. They had no previous knowledge about object-oriented programming.

60 10th graders of two different classes from a German secondary school were asked two different questions before and after they had learned sequence diagrams. First, they had to answer how the order of the following java statements could be changed:

```
1: TrafficLight light1;
2: light1 = new TrafficLight();
3: light1.setPosition(1,0);
4: light1.setColor("red");
```

Then they had to analyze two different sequences of statements, which contain the same statements in different order. The students had to justify if both sequences were correct and whether both sequences were leading to the same results or not.

The students had to justify in written form which of these statements could not be switched.

```
1: TrafficLight light1;
2: light1 = new TrafficLight();
3: light1.setPosition(1,0);
4: light1.setColor("red");
```

IV. ANALYSIS

All answers were analyzed using the qualitative content analysis [13] of Mayring. Therefore, sentences of students were rewritten, e.g. by transforming colloquial expressions to High German. This step is called “paraphrasing”. “A paraphrase is the content of a phrase without any decorative or filler words, it is the core meaning of the phrase. The semantic content is equivalent to the phrase, but is expressed in a short form” [13].

The paraphrases are often formulated on a uniform stylistic level. For example, if a student wrote “hmm.. line 3 and 4 could be switched, hmm.. cuz order not matter” it would be paraphrased as “Line 3 and 4 could be switched, because the order does not matter” It is important to mention that the paraphrasing depends on the research question. If the researcher wants to analyze the hesitation of the students or how convinced the students are, then of course it is not allowed to remove the “hmm” from the original phrase in the paraphrasing step. For analyzing hesitation, self-corrections [14] or feelings qualitative content analysis would not be the first choice. In these cases discourse analytic approaches would fit better, like Conversation Analysis [15, 16] introduced by Sacks, Schegloff and Jefferson. Much more problematic is a student phrase like “line 2 and 3.. hmm.. no, line 3 and 4 could be switched” in this case we have a clear “self-correction” [14] and this could not be paraphrased loss-free using a qualitative content analysis [13] which does not derive from a discourse analytic approach. In the following section (V. RESULT), we are going to present two students’ phrases, which would lose information in a paraphrasing step. A Conversation Analysis [15, 14, 16] would have been better in their case.

A further alternative to qualitative content analysis [13] for analyzing student answers would be a modern hermeneutic approach, like Grounded Theory Coding [17]. However, this
approach of content analysis is open to every kind of content that a text contains and thus it could easily uncouple the interpreted text content from our research question [18]. Discourse analytic approaches have the same problem [13] as they are often based on modern hermeneutic approaches [18]. Therefore, the qualitative content analysis of Mayring is the best choice for our analysis.

As the example set is very small (about 60 participants) and is not representative (two classes from the same school and country) no statistical conclusion can be made. Calculation of percentage, standard deviation, and significance would be misleading and deceptive (see also section VII. DISCUSSION). Instead of a quantitative angle of view, we decided to choose a qualitative view and are interested in existence propositions. Are there students who change their conceptions of statement order after the introduction of sequence diagrams?

V. RESULTS

Some students do not see any problems by switching statement 2 and 3 or 2 and 4 of the first question before sequence diagrams were introduced and practiced. Unfortunately, this interchange of statements would cause a null pointer exception in java. (The students would not notice this error until runtime.) This means that in their conception it is possible to call a method of an object before the object is instantiated or the instantiation was already done by declaration (statement 1) or will be done by calling the first method of an object (statement 3 or 4). Student No. 15 “It is possible to switch 2, 3 and 4. I can’t be switched, because in Line 1 the name of the object is written.”

Student No. 09: “It is allowed to switch 2, 3 and 4, because first an object has to be declared, after this, the creation of an object is possible. The order of changing attribute values does not matter at all.” This student is arguing correctly, but it seems that Line No. 2 is also interpreted as “changing attribute values”. Maybe because the creation of an object is some kind of setting the “value” of a reference attribute. Student No. 17 “1 must be at the first place, cause of class definition, the rest could be arbitrarily switched.” However, not every student answer is incorrect. Student No. 03: “1 and 2 must be on same place, because in the beginning an object needs a name and then you can create it. 3 and 4 are exchangeable, because it doesn’t make any difference, whether the position or the color is changed first.”

After the introduction of sequence diagrams (one lesson) and the exercise (another lesson), the students changed their answers, e.g. Student No. 15: “3 and 4 could be switched. First you have to declare the traffic light, then you have to create it. Which attribute is changed later, does not matter.” And No. 17 answered: “1 and 2 cannot be switched. But 3 and 4 can.”

Unfortunately, no student argued using explicitly sequence diagrams for their answer of the first question. Therefore we cannot claim a direct connection between teaching sequence diagrams and correct answers for the first question. We are just able to notice a positive change of answers. Luckily, the argumentation for the second question unveils new misconceptions in the beginning and is clearly influenced directly by sequence diagrams in a positive way.

Before introducing sequence diagrams many students answered to the second question that only sequence 1 was correct and that sequence 2 had a wrong order of statements. Student No. 06: “At the beginning an initialization is necessary and only then it is possible to call methods. Therefore just sequence 1 is correct.” This student answered the first question correctly, but analyzing answers of the first question would never lead to finding the misconception the student explained answering the second question. This student states that in algorithms the initialization of all objects should precede all object calls. Even if object 1 is instantiated, it would not be correct to call a method of object 1, as long as object 2 is not instantiated, as well. Student No. 19: “Sequence 1 is correct, sequence 2 leads to an error. Therefor they do not have the same result. Both are creating lights before working with them, but all lights have to be defined from the beginning. That’s why sequence 2 is wrong.” Quite similar to student No. 19 and No. 06 are the argumentations of students No. 02-06, 08-09, 14, 20-22, 24-27 and 29. Let’s call this misconception: “Every object of a program has to be instantiated right at the beginning.”

The rest answered correctly, some without explanation and some with, like Student No. 50: “I say both sequences are correct, because the statement order in a sequence does not matter. Both lead to the same results.” (Here it is unclear if the order does not matter in general or just does not matter in the given example.)

Very interesting it the answer of Student No. 19 after sequence diagrams had been introduced. The student changed her answer and argued using sequence diagrams: “1.) 1st sequence: it is correct. In the beginning, the lights are created and then moved to the correct position and so on. 2nd sequence: it is also correct -> both lights were created separately -> it would be possible to draw a sequence diagram; one diagram would be higher. 2.) Both are leading to the same result, because they are containing the same instructions”. Having had a misconception at the beginning, this students later analyzed the two sequences correctly and argued that the second one was also correct, because of the related sequence diagram. If it is possible to draw a sequence diagram, the sequence of statements will be possible as well. Student No. 29 argued similarly: “1.) Both sequence diagrams are correct, because both are drawing the objects first. Both sequences are correct. 2.) Same result as the objects have the same attribute values”. Student No. 13: “Sequence 2 is correct according to its sequence diagram. Sequence 1 not. Results are equal.” This student also argued using sequence diagrams. However, he now makes a mistake interpreting the first sequence without explaining it.

Many students, who had made mistakes and had answered incorrectly before the introduction of sequence diagrams are now giving the correct answers, but do not mention sequence diagrams in their argumentation, like students No. 2-6, 8-9, 2022, 24-26 and 27.

These results clearly show a positive influence of teaching UML sequence diagrams to students’ concepts of statement order.
At last, we want to show two student answers which are difficult to paraphrase using our quantitative content analysis approach [13]. Both answers are to the first question after the introduction of sequence diagrams. Student No. 31: “it is possible to switch line 3 and 4, because it does not matter if the position or the color is changed first. However, in first place the object has to be instantiated declared and after this, it has to be instantiated.” This is a self-correction [14] and directly refers to the order of object declaration and instantiation. Student No. 32: “You must [can not] switch line 1 & 2”. (This student eliminated "must" and inserted "can not". In the German phrase the word "nicht", which stands for "not", appeared at the end of this sentence, therefore it is clear that he changed the meaning of this sentence from "you have to change" to "it is not possible to change"). Both phrases should illustrate the limits of the chosen content analysis.

VI. CONCLUSION

We analyzed the influence of teaching sequence diagrams to conceptions and misconceptions of computer science students in a qualitative [13] way. We observed a positive influence, like the disappearance of some misconceptions and problems, which are connected to the statement order. We discovered a misconception ("Every object of a program has to be instantiated right at the beginning.") and showed that sequence diagrams are a good way to enrich students’ conception of statement order.

Sequence diagrams are not limited to help only university students with thread and synchronization problems [7], as had been described in literature before. They also could help students of secondary schools with basic object-oriented programming problems and are definitely more than just modeling techniques.

VII. DISCUSSION

Our study was just a small (qualitative) study; we analyzed the arguments of 60 students answering two questions about statement order from two different classes from the same country, therefore we can just claim existence propositions about observed problems and misconceptions and the positive influence of teaching UML sequence diagrams on the very same problems and the very same students. To predict a good influence of early teaching sequence diagrams in general we need much more students of different schools from different countries and we have to broaden our view to more misconceptions, conceptions and possibly also to competencies. Only then statistical significance could be computed and other influence variables could be systematically excluded.

For some student answers, another content analysis, e.g. Conversation Analysis [15] would fit better, like for the phrases of students 31 and 32 (see last paragraph of section V). However, the given answers were in written form. Therefore, the teacher was not able to ask why they had changed their answers. Only interacting with the students, e.g. in an interview, would lead to enough information for a full and sufficient Conversation Analysis, but would also lead to problems with quality criteria due to these interactions.

VIII. LITERATURE