An empirical study on the impact of introducing a modeling tool in a Requirement Engineering course

Loli Burgueño
Open University of Catalonia
Barcelona, Spain
lburguenoc@uoc.edu

Javier Luis Cánovas Izquierdo
Open University of Catalonia
Barcelona, Spain
jcanovasi@uoc.edu

Elena Planas
Open University of Catalonia
Barcelona, Spain
eplanash@uoc.edu

***Abstract*—In numerous Programming and Software Engineering courses, students are asked to program on paper. This has supporters and detractors. Among its advantages, supporters claim that programming on paper allows students to focus on functionality, avoiding the distractions caused by syntax and without limiting their thinking to a specific programming language or paradigm. Detractors claim that this method lacks advanced capabilities provided by IDEs such as syntax check and auto-completion. More importantly, it does not give the opportunity to execute and test the code, which prevents students from discovering bugs.**

The state of the art has studied the benefits and disadvantages of programming on paper versus computer for general-purpose languages like Java and C with students of initial courses. Nevertheless, to the best of our knowledge, no study has been done targeting formal languages like OCL, which are taught in advanced courses.

In this paper, we present our experience after introducing a modeling tool for the specification of OCL constraints in a Requirements Engineering course. This course is optional and is offered in the third and fourth years of the Computer Engineering degree. Our study covers two academic years, 2019 and 2020, in which there were 136 and 161 students enrolled, respectively. We present the context and design of our experiment, the results obtained from the empirical study we have performed and our conclusions, which support the suitability of the use of tools.

***Index Terms*—Requirement engineering, modeling tools, OCL, teaching, empirical study**

I. Introduction

In recent years, the professors of the Requirements Engineering course of the Computer Engineering Bachelor Degree offered at the Open University of Catalonia (Universitat Oberta de Catalunya–UOC for short) have noticed that students showed some disappointment with the formative assessment. As part of this formative assessment, they were asked to define a series of restrictions using the declarative language OCL [12]. No digital support was recommended or provided and the students used to do this exercise on paper. The complaints of some of the students, added to the feeling that the lack of a software tool could be affecting the development and results of this exercise, made us consider the introduction of a modeling tool. The idea behind this change is that the tool could allow the students to execute and test such OCL constraints and therefore we could be preventing a potential learning obstacle.

The use of paper in Programming and Software Engineering courses has its supporters and its detractors. On the one hand, many educators endorse the benefits that programming on paper has for Computer Engineering students or for students of any other higher education program. They claim that programming on paper allows students to focus on the logic of the program they are writing, avoids distractions caused by syntax errors, and does not limit the students' thinking to a specific programming language, platform or paradigm[1]. These statements are also explicitly supported by many international companies. For example, during the hiring process, one of the several interviews that their interviewees must pass is the so-called *whiteboard interview*. During these sessions, the potential employees need to solve a problem on a whiteboard using the (pseudo-)language they prefer. This way interviewers assess the knowledge, competences and skills of a potential hire.

At universities, pencil-and-paper programming is frequently used not only as a means of learning but also when assessing the students' knowledge in both formative and summative assessments [1], [19], [20].

Despite its benefits and adoption in specific contexts, paper programming also has its detractors and some widely recognized drawbacks. For example, students often complain that they cannot execute, test, and debug the code they are writing. This method prevents students from verifying correctness of their code and hinders the detection of those errors that could be easily and early discovered with the simple execution of the program.

Apart from the studies that address programming using general-purpose languages (e.g., [1], [19], [20]), to the best of our knowledge, there is no study that focuses on the use of paper as opposed to a tool when learning formal languages and/or standard languages for the definition of rules (such as OCL). Since modeling languages such as UML/OCL [12], [13] are extensively used in the academic environment, not only in our university but in many others[2], we decided to document and publish our experience so that anyone in our situation can benefit from it.

Our study shows that the use of tools for learning rule-definition languages such as OCL is perceived positively by

[1]https://classcube.com/write-code-paper/
[2]https://mde-network.github.io/#miembros

both students and faculty members. Comparing to the results obtained in the fall of 2019—when no modeling tool was provided to our students—and the fall of 2020—semester in which we introduced a modeling tool and performed this study—, we have observed that the students' grades have improved.

The rest of the paper is structured as follows: Section II explains the context of the course where the experiment was carried out; Section III describes the experiment and the methodology we followed; Section IV presents the results obtained; Section V presents the threats to validity; in Section VI we present our discussion about the results; Section VII explains the related work and, finally, Section VIII briefly presents our conclusions and several lines of future work.

II. Context

The experience described in this paper takes place in the Requirement Engineering course of the Computer Engineering Bachelor Degree offered at the Open University of Catalonia (UOC). Our Bachelor programmes have 240 ECTS credits and are planned to take four years of full-time study. This is an elective course that students can take during their third or fourth academic year. The course assumes a basic knowledge of software engineering and delves into the first stage of the software development life cycle. The contents of the course are organized into five modules: (1) introduction to requirements engineering; (2) requirements elicitation; (3) requirements analysis and management; (4) requirements documentation; and (5) requirements validation and verification. The OCL language is introduced during the fourth module, as a method to formally document requirements.

The progress of our students in this course is evaluated using a continuous assessment model. Concretely, the model of the course is composed of four Continuous Assessment Tests (CAT) scheduled throughout the semester, all of them formative. The course does not have any summative test. For each CAT, students are provided with detailed feedback consisting on their grade accompanied with individual comments. A few days after the feedback is given, we publish the solution to the CAT.

All CAT activities are built on top of the same case study, which allows the students to have a complete and more realistic vision of all the phases of requirements engineering lifecycle. In particular: the first CAT (CAT1) focuses on requirement elicitation given the textual description of the case study; the second CAT (CAT2) addresses the analysis and management of requirements; during the thrid CAT (CAT3), the students document the requirements in an agile way through use cases and user stories; and, finally, in the fourth CAT (CAT4), the students document the requirements in a formal way (using UML/OCL) and apply validation and verification techniques. CAT4 is the target of our experiment.

To give the reader more details about the content and scope of the fourth CAT, let us explain that the CAT4 in the last edition of the course (fall semester of 2020) consisted of formally documenting and validating the requirements of a social network (see Question 14 in [10][3]). In particular, given the class diagram in Figure 1, students were asked to document four integrity constraints using OCL.

For illustrative purposes, let us show some of the constraints the students had to define:

> *Constraint #1*: "Videos can only be watched (`viewedDate`) once they have been uploaded to the platform (`updated`) and as long as the user that published it is registered in the platform (`subscriptionDate`)".

One of the possible OCL constraints to this question is:

```
context View inv:
  self.viewedDate >= self.video.uploaded and
  self.viewedDate >= self.user.subscriptionDate
```

> *Constraint #2*: The number of different users who have viewed each video must be less than or equal to the number of total users in the system. Keep in mind that, if a user has viewed the same video several times, we want to count it only once. Remember that you can calculate the number of total users in the system as follows:
> `User::allInstances()->size()`.

One of the possible solutions to this constraint in OCL is:

```
context Video inv:
  self.views->oclIsUndefined() or
  (self.views->collect(d | d.user)->asSet()->size()
  <= User::allInstances()->size())
```

The faculty involved in this course is formed by two assistant professors (the two first authors of this paper) and three teaching assistants. The course has around 150 students enrolled each semester, which are assigned to virtual classrooms with about 70 students each. Each classroom is energized by one of the teaching assistants, who guides and accompanies the students during their learning and assesses them and returns the correspondent feedback for each activity.

During the semester, all communication is carried out asynchronously and online, through the virtual classroom forums (where the messages are public to all students in the classroom). Less often, students communicate through email to ask questions directly and privately to the teaching assistants.

III. Experiment description

We built an experiment to address the following research question:

RQ. *What is the impact in the student learning process of a modeling tool with support for defining and executing OCL constraints?*

We propose to measure the learning process by relying on three indirect measurements, namely: the student's self-assessment, the teaching assistants' assessment and the student's academic performance.

Our hypothesis is that using the tool will have a positive impact in the understanding and learning of the OCL language, as the ability of experimenting during the learning process allows understanding the syntax and semantics of the language,

[3]Materials in Spanish.

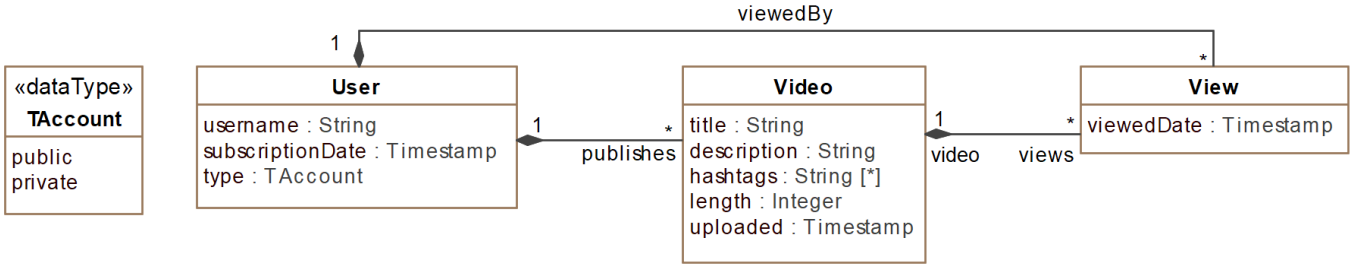

Fig. 1. Class Diagram corresponding to the CAT4 in the fall of 2020.

detecting errors (i.e., enabling the test-and-error behavior); thus, promoting the definition of correct constraints.

Our research context is a case of Action Research. We employed empirical research methods [7] to answer our research question. In particular, we have performed a mixed-methods study combining qualitative and quantitative research approaches to address our question from different perspectives with the goal to mitigate the weaknesses of the different empirical methods used.

We defined a controlled experiment to study the cause-effect relationship between using the modeling tool (i.e., binary independent variable) and its consequences. We identified the following dependent variables: dedicated time to do the experiment, student's perception on the usefulness of the modeling tool, student's perception on the difficulty of use of the tool, and student's academic performance. Next we describe each variable.

The experiment was done during the CAT4 of the course, which is composed of three exercises. One of the exercises assesses the knowledge about OCL and weights 40% of the total mark of the CAT. This exercise presents a class diagram and includes four question tests which ask the student to define OCL constraints (as stated in the previous section). To facilitate the analysis and comparison of the results, the structure of the CAT is similar to those used in CATs from previous semesters. Thus, the OCL constraints to define in the exercise evaluate the same OCL features (e.g., variables, operators, traversals, etc.).

Prior to start the CAT, students were informed about the experiment and were given the opportunity to choose between using or not the modeling tool to solve the CAT's exercises. The final decision to use the modeling tool is therefore up to the student. The modeling tool proposed was MAGICDRAW [16], as it is one of the most popular UML modeling tools with support for the definition and execution of OCL constraints. Besides, MAGICDRAW is used in the Software Engineering course, which is a prerequisite for Requirements Engineering course.

Once started the CAT, students had four weeks to work and deliver their solution proposal. During this development time, students could ask questions either in the virtual classroom's forum or contacting the teaching assistants of the course via email. After this period of time, we collected the experiment's data from three sources, namely: (1) students, via an online form; (2) teaching assistants, via online structured interviews; and (3) messages and grades, via the university virtual campus.

Next we describe the data collected from each source.

A. Student Questionnaire

In our experiment, we collected students' data via anonymous online forms. To create these forms, the authors of this paper discussed and worked in several versions. Once we reached a final version, it was validated with the teaching assistants.

Forms were composed of two sections. The first section included general questions to perform student profiling and know her/his knowledge of OCL prior to the experiment. Thus, this section contained demographic questions and collected the student's grade in the Software Engineering course (i.e., prerequisite for our course). We also asked general questions about OCL that allowed us to know whether the student knows the language, her/his perception (i.e., whether it is an easy or hard language to use) and whether she/he thinks that tool support is required to learn the language.

The second section of the form was composed of questions specific to the OCL exercise of the CAT, which varied depending on whether the student used MAGICDRAW or not. If the student used the tool, the forms included questions to know the reason for choosing to use the tool, the student's experience with the tool, time required to solve the exercises and the student's perception on the difficulty to solve each part of the exercise. Additionally, there are questions to know whether the tool allowed students to practice with OCL beyond the constraints required to solve the exercises, thus promoting self-learning. If the student did not use the tool, the form included questions to know the reason of this choice, time required to solve the exercises and the student's perception on the difficulty to solve each part of the exercise. The form can be downloaded from [10].

B. Structured Interviews with Teaching Assistants

To collect comments and opinions from the teaching assistants, we performed a set of structured interviews.

We interviewed the three teaching assistants, who were in charge of assessing the CAT and answering questions in the classrooms. The interviews were structured according to four

areas: (1) general view, where teaching assistants provided their perception on how MAGICDRAW may have affected the CAT; (2) her/his opinion on whether we should keep using the tool in future semesters; (3) the degree of interaction with students during the development of the CAT, which would let us know whether introducing a tool affected the communication between the students and the teaching assistants (e.g., whether the teaching assistants had to solve technical questions about the usage of MAGICDRAW and not related to OCL itself); and (4) conclusion and additional comments, in order to identity possible improvements for future semesters.

C. Messages and Grades

To complement our evaluation, we collected the number of messages in the virtual forums and asked the teaching assistants to share with us information about the emails they exchanged with the students related to OCL. We also collected the students' grades for the CAT4 as well as the course final grades. Additionally, we also collected the students grades from the previous semester (Fall 2019), which enables a comparative analysis.

IV. EMPIRICAL EVALUATION

A. Data Collection

As mentioned in Section III, the data collection took place after the students finished the CAT4 and the data comes from three different sources: (1) the questionnaire that we asked the students to fill during the first week of January 2021; (2) the interviews with the teaching assistants at the end of January 2021, once the had finished their duties in the course and had evaluated and graded the CAT; and (3) the messages exchanged in the forums and the grades obtained in the Fall of 2020 that were collected at the beginning of February 2021.

B. Results

Before jumping into analysing the data, we have validated that all the the forms have been filled out correctly and there was no need to discard entries (e.g., due to improper or out of scope answers).

1) Students Questionnaire: The participation in the questionnaire was optional and 32% of the students responded (i.e., 52 out of the 161 students enrolled in the course). For informative purposes[4], we report on the demographic data that we asked our students. Out of these 52 students, 46 identified themselves as male, 6 as female, and there were no students who identified with 'other'. Only one student is in the age group between 18-23 years old, 41 students between 24-45 years old, and 10 students above 45.

General Questions. First of all, let us report on the results from the general questions section.

To the question about whether they knew OCL prior this course, 75% of the students replied no, 19% said that they had heard about it and only 6% (3 students) said that they had some knowledge about OCL. It is worth noting that these 3 students had not been enrolled in the course before, hence they must have learned and used OCL in another context outside this course.

[4]Note that these are not dependent variables in our experiment

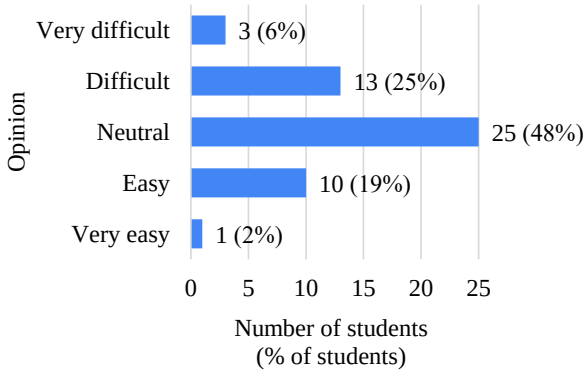

Fig. 2. General perception of OCL.

Then, we asked the students about their general perception of the OCL language and, in particular, about the level of difficulty of OCL. We used the 5-point Likert scale that ranges from very complex (-2) to very easy (+2). The results are those shown in Figure 2. Both the mode and the median of their responses were 0, meaning that they neither agree nor disagree. The average was -0.13 with a standard deviation of 0.86 points.

Note that, despite the quantitative nature of the Likert scale, which normally translates to integer values between 1 and 5, we have opted to translate this scale to a decimal scale in the range [-2..2]. This way, apart from modes and medians, we compute averages and standard deviations and interpret positive numbers as positive results and negative numbers as negative results. Our university uses this strategy to obtain more precise and informative data when analyzing student satisfaction surveys.

In the questionnaire, we asked the students about their grade the Software Engineering course (which is a prerequisite of the Requirement Engineering course). Using the same Likert scale as before, we analyze their opinions about OCL, but this time grouped according to the grade that such students obtained in Software Engineering. The results are shown in Table I. We can observe a direct relationship between Software Engineering grades and students' perception: students with higher grades tend to think that OCL is easier, while students with lower grades found it more complex.

We also asked our students whether they considered that learning OCL requires to follow the trial and error method and therefore, needs tools. Once again we used the Likert scale of agreement. We show the results in Figure 3. The statistic analysis of these responses show that a mode is 2 (strongly agree), the median is 1 (agree), and the average is 1.04 with a standard deviation of 1.03. This indicates that students tend to think that the use of tools is beneficial if not necessary.

Table II shows the statistical results for this same question considering independently the group of 17 students who did

Grade in SE	Num. of Students	Mode	Median	Average	Std.
A+	2	-	0	0.50	0.71
A	3	0	0	0.33	0.57
B	27	0	0	0	0.88
C	7	0	0	-0.57	0.79
Validated	13	0 & 1	0	-0.38	0.87
Total	52	0	0	-0.13	0.86

TABLE I
PERCEPTION OF THE DIFFICULTY OF OCL WITH RESPECT TO THE GRADE OBTAINED IN SOFTWARE ENGINEERING.

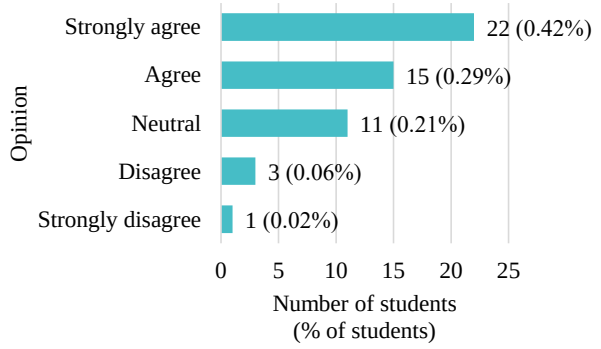

Fig. 3. Responses to the question about whether tools are needed to learn OCL.

not use MAGICDRAW (control group) and the group of 35 students who did use it (experimental group). The average of the opinions shows how even those students who did not use MAGICDRAW tend to think that its use is more necessary than unnecessary (0.41, which is a positive number).

Regarding the time spent on solving the OCL exercise, the students who used MAGICDRAW reported that it took an average of 6.11 ± 4.61 hours, while the students who did not use it needed 5.56 ± 4.84 hours. Figure **??** shows graphically these data. To statistically compare if there is a difference in the time it took for both groups, we have done the Welch t-test with two tails and unequal variances. Our null hypothesis

	# students	Mode	Median	Average	Std.
With MD	35	2	2	1.34	0.84
Without MD	17	0	0	0.41	1.12
Total	52	2	1	1.04	1.03

TABLE II
RESPONSES TO THE QUESTION: "I THINK THAT LEARNING OCL NEEDS TOOLS TO BE ABLE TO DO TRIAL AND ERROR" USING THE LIKERT SCALE OF AGREEMENT DEPENDING ON WHETHER THEY USED MAGICDRAW (MD).

	With MD			Without MD		
Exercise	Average	Mode	Median	Average	Mode	Median
A	0.829	2	1	0.29	1	0
B	0.971	1	1	0.18	0	0
C	0.829	2	1	0.18	0	0
D	-0.400	-1	-1	-0.65	-1	-1

TABLE III
DIFFICULTY OF EACH OCL EXERCISE IN THE CAT.

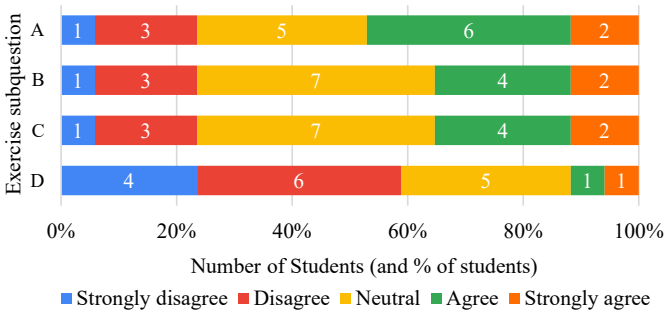

Fig. 4. Difficulty each OCL exercise in the CAT according the students that did not use MagicDraw.

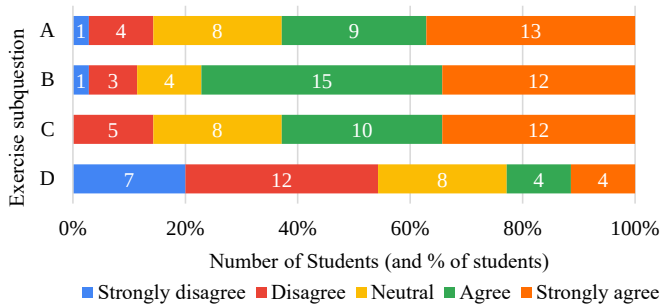

Fig. 5. Difficulty each OCL exercise in the CAT according to the students that used MagicDraw.

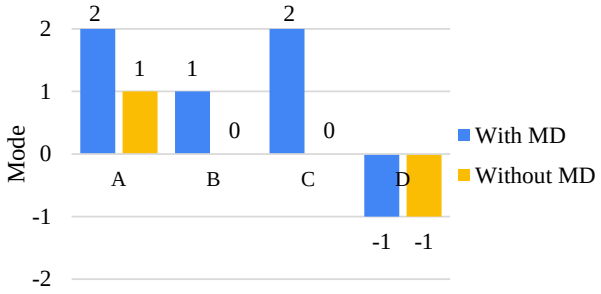

Fig. 6. Students' perception of the difficulty of the OCL exercise (Mode).

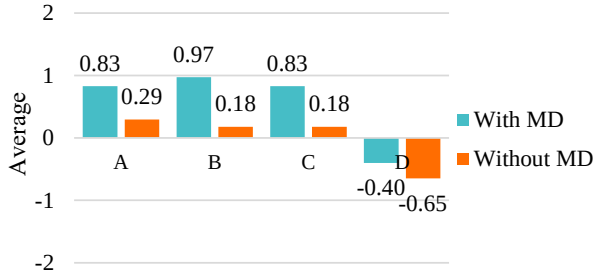

Fig. 7. Students' perception of the difficulty of the OCL exercise (Average).

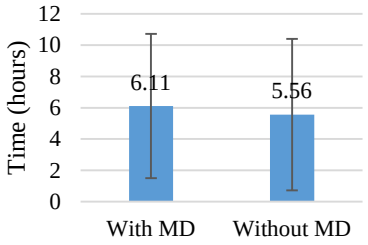

Fig. 8. Time spent by the students solving the OCL exercise (average and standard deviation).

is that the times between both groups do not differ. The t-test gives a value of 0.71. Since this value is greater than $\alpha = 0.05$, we accept the null hypothesis and conclude that there are no significant differences associated to the use of a tool when solving the OCL exercise.

Specific questions for students not using MAGICDRAW. Here we show the results to the specific questions to students who did not use MAGICDRAW.

We asked them what was the reason for choosing not to use MAGICDRAW. Five of them said that they "considered that they already knew how to write constraints and did not need to use a tool", 6 of them said that "they could not make MAGICDRAW work", 4 said they had not used it "because they did not want to waste time using a tool", and 2 people said that it was due to the "lack of time". Let us say that several students complained about MAGICDRAW being incompatible with the latest macOS update, which is a problem that MAGICDRAW had not faced in the past. Interestingly, out of the 5 students who said that they knew how to write OCL without a tool, 3 had used MAGICDRAW before, 2 thought about using at the beginning but but they later discarded the idea, and one said that the tool is difficult to use and that she/he did not consider that it could provide a substantial benefit.

With respect to the perception of the difficulty to solve each one of the subquestions of the OCL exercise (subquestions A–D), we asked the students using the question: "I found that solving the exercise *X* was easy...", and again, we collected their answers using the 5 point Likert agreement scale, which goes from strongly disagree to strongly agree. Figures 4 and 5 show the students' opinions for each question grouped by whether they used or not MAGICDRAW, respectively. Then, we translated the results to the range [-2..2] and computed for each question the average, mode and median of the opinions. The results are presented in Table III and graphically represented in Figures 6 and 8.

Figure 6 shows in blue (bar on the left) the mode of the responses provided by the students who used MAGICDRAW, and in yellow (bar on the right) those who did not use it. Figure 8 shows in turquoise (bar on the left) the average of the answers given by the students who used MAGICDRAW and in orange (bar on the right) the average of the responses from those who did not use it. We can observe how, in general, all the students considered that exercises A, B and C were easier than exercise D. However, we can appreciate that those students who used MAGICDRAW found that the exercises were easier than those students who did not use it.

Specific questions for students using MAGICDRAW. Lastly, we present the results to the specific questions we asked students who used MAGICDRAW. Using the Likert scale again, we asked students:

- whether MAGICDRAW had helped them solve the exercise faster. The mode and mean of their responses were both 0, which indicates that the students did not consider that MAGICDRAW could have had any impact on the time that they had to invest into solving the exercise;
- whether MAGICDRAW had allowed them to solve the exercise more easily. In this case, the mode is 1 and the average 0.54, which indicates that students tend to agree with the fact that the tool helps to make their task easier;
- whether MAGICDRAW allows them to learn OCL in practice and enables them to do more realistic and complex exercises. For this question, the mode was 0 and the average 0.63, which indicates that students tend to agree with this statement;
- whether they would recommend using MAGICDRAW in the Requirement Engineering course in the future. The students' response mode was 1 and the average 0.86. This means that, in general, the students recommend the use of MAGICDRAW in the future.

We also asked the students:

- whether they thought that using MAGICDRAW had allowed them to experiment with OCL constraints beyond those required in the exercises. To this question, 21 students (60%) said yes, while 14 (40%) said no;
- whether the use of MAGICDRAW had allowed them to solve their OCL questions and doubts autonomously and without having to resort to the teaching assistants or the classroom forums. 23 students (66%) said yes, 9 (26%) said no, and only one student claimed that he had no doubts at all.

For the sake of transparency and in order to allow the reproducibility of our study, both the questionnaire and the students' responses can be found at [10].[5]

2) Structured interviews to the three teaching assistants: The three teaching assistants, and especially two of them who were part of the Requirement Engineering course both in the fall of 2019 and the fall of 2020, agreed that introducing MAGICDRAW was positive. They encouraged us to continue giving the students the option to use a modeling tool.

They said that there had not been many questions regarding the use of MAGICDRAW, beyond some installation issues on macOS BigSur.

When we asked them whether introducing the tool had implied a greater workload for them, their answer was no.

[5]Material in Spanish.

Course	Number of students	Forum Messages	Emails	Messages/ Student
Fall 2019	136	35	20	0.40
Fall 2020	161	61	70	0.81

TABLE IV
STUDENT-FACULTY INTERACTION

Course	Num. of students	Not taken	CAT Grade	Course Grade	Weighted CAT Grade
Fall 2019	136	6	8.08±1.14	7.91±1.20	6.39
Fall 2020	161	9	8.46±1.15	7.92±1.29	6.70

TABLE V
CAT GRADES FOR THE FALL OF 2019 AND 2020.

They also told us that they sensed that the students' questions in the fall of 2020 were more specific, and that they seemed to have a better idea of how to deal with the exercises. They said that in the previous semesters, the students seem to be more confused.

As an additional comment, one of the teaching assistants told us that he had noticed that the quality of the solutions had increased. It caught his attention that the students were using the OCL syntax more correctly. He pointed out that, in previous years, some students tended to write queries in OCL with an SQL-like syntax, whereas this semester he found that this did not happen as often.

3) Comparative fall 2019 and fall 2020: Table IV collects the results on the interaction between teachers and students in forums and by email. In the last column we can see how, for this CAT, the ratio of messages per student in 2019 was 0.40 while in 2020 it increased to 0.81. This reflects that in 2020 there was more activity in classroom forums and more interaction with the teaching assistants.

In Table V, we have collected the data that enables the comparison between the students' grades in the fall of 2019 and 2020. For each semester, it shows the number of students, the number of those students who did not submit the CAT (column Not Taken) and the average and standard deviation of the CAT grades. To test whether, statistically, the grades in both courses are significantly different, we have performed a Welch's t-test with two tails and equal variances. Our null hypothesis is that there is no difference in grades. The test provides a value of 0.0065. Since $0.0065 < \alpha = 0.05$, we reject the null hypothesis. The t-test has confirmed that grades in both courses have been significantly different.

Despite having created similar CATs in both years, we are aware that there might be different reasons that led to differences in the grades in both semesters. Therefore, in order to make a comparison as fair as possible, we have also collected the final grades that the students obtained for the course in both 2019 and 2020. Then, we have weighted the CAT grades with respect to the final grades. Looking at the last column of Table V, we can observe that the weighted grade in 2019 was 6.39, while in 2020 it was 6.70. Given that the average in 2020 is 0.31 points higher, we can conclude that the CAT grades were higher in 2020.

Despite the COVID-19 pandemic that affected our course in the fall of 2020, the online nature of our course in all its editions considerably reduced the impact. The pandemic did not change anything with respect to the logistics of the course. Nevertheless, we were aware that our students and teaching assistants could have been experiencing difficulties. Therefore, in 2020, we decided to be more flexible with the deadlines and grant a few extra days upon request. No one used these extensions.

V. THREATS TO VALIDITY

A. Internal validity

Internal validity checks whether the test or instrument measures what it is supposed to. This threat can affect the independent variable with respect to causality. That is, the results may indicate a causal relationship, although there is none.

In this respect, we are aware that empirical methods are not infallible and various factors can influence their validity. For instance, ignoring a variable that may have an impact on the results, and therefore on the conclusions. To mitigate this threat we have used more than one method to answer the same question, including both quantitative and qualitative methods: questionnaire to students, structured interviews with teaching assistants, and a quantitative comparison of the grades and interaction between students and faculty both semesters.

The questionnaire that we provided to the students was anonymous. This prevented us from doing a deeper analysis such as relating the use (or not use) of MAGICDRAW with the final grade of the students. However, we believe that our decision was the most appropriate since anonymity improves the honesty of responses.

We let our students choose whether they prefer to use MAGICDRAW or paper. Although this introduces a clear selection bias that may affect the internal validity, our students have the right to choose. In a case of Action-Research like ours, this threat is hardly avoidable without compromising fairness.

Asking students whether they recommend the use of MAGICDRAW after using it or not can lead to a biased response. Nevertheless, this was unavoidable in a scenario of Action-Research like this. Cross-validation using different empirical methods helps mitigate these biases.

Two of the three teaching assistants were part of the course in both semesters (fall 2019 and fall 2020), and only one joined in 2020. This ensures that at least these two instructors have evaluated and interacted with the students in the same way. In addition, they are in a position to express their personal opinion and experience on the development of both semesters.

Our goal was to make the comparison between the semesters in 2019 and 2020 as fair as possible. Therefore, although describing a different use case, the CAT4 in 2020 tried to faithfully mimic the structure and content of the CAT in 2019. Let us say that the CATs of previous years are not published and, in theory, our students do not have access to them. The

similarity between the two CATs should not have altered the grades. In addition, when comparing the CAT4 grades obtained in 2019 and 2020, we have computed the weighted CAT grade with respect to the final course grade to mitigate the impact of the comparison of different CATs and different students.

B. External validity

This kind of threat limits the ability to generalize the results beyond the experiment context.

In this respect, in the context of our study, external validity can be threatened due to the fact that our experiment takes place in a real environment where students must have the option to choose whether to use MAGICDRAW or not. Since we could not make a fair and random assignment, this resulted in unbalanced groups which might have affected the statistical results. However, given that the number of students in each group is significant, we believe that the results can be generalized to other courses and subjects where the definition of OCL constraints is addressed.

VI. DISCUSSION

In this section we discuss the results presented in previous section.

Modeling tools help on the definition of OCL constraints. We have observed that students preferred to use modeling tools to address the CAT. Our results show that student's perception about the difficulty to solve the exercises is lower when they employ the tool.

Modeling tools foster experimentation and self-learning. Students that decided to use MAGICDRAW admitted that the tool gave them the freedom to experiment with OCL constraints and practice self-learning. Besides, they would also recommend using this kind of tools in other courses.

Student's performance is slightly better. Although our form was anonymous and there was no relationship between the grades and the student's identity, the final grade of those students who used MAGICDRAW was better. However, we believe it is necessary to perform a more exhaustive study, with additional experiments in future courses, to confirm that this difference is significant, as current results may be linked to external factors not considered in our study.

No technical barrier to entry. In our experiment with students, we did not detect troubles related to the installation or use of MAGICDRAW other than the issues reported when installing the tool in the latest macOS version, which we easily addressed by providing a virtual machine. Note that MAGICDRAW has never experienced any problem in macOS before, however, the last version (codenamed *BigSur*) has caused several issues with, not only MAGICDRAW but a number of software applications. Teaching assistant also supported the use of modeling tools and recommended its inclusion in future courses.

Teaching assistant's workload was not affected by the use of modeling tools. Although the number of messages between students and teaching assistants was higher in 2020, the latter did not detect any increase in the workload. We believe that it may be due to the fact that students' questions were more concise and easy to address.

Low impact in classroom organization. As the use of MAGICDRAW was optionally, it did not imply any structural change in the organization of the course. However, as a result of our experiment. we may explore the modification of the course to make the use of modeling tools compulsory, thus allowing us to expand the course objectives.

Introducing new tools require extra effort from coordinating professors. To introduce a new modelling tool coordinating professors had to perform an explorative study of existing tools, prepare manuals and documentation. The latter is specially important in MAGICDRAW, as its features vary according to the version and license provided.

Limited number of tools supporting OCL validation and verification. Flexibility to propose and use different modeling tools is limited as there are very few with full support for OCL (i.e., including validation and verification). In our case, we decided to use MAGICDRAW due to its friendly interface and the fact that students were already familiar with it, but it is a proprietary tool. We also considered USE [9], whose use we are currently exploring for the future. Another problem was the lack of reference examples (and their implementation in the corresponding modeling tool) [8].

VII. RELATED WORK

The use of software development tools, both in professional and academic environments, has been widely studied in the literature. Many of these studies are based on experiments that compare several features of development tools in order to determine their usability. Some examples in the academic field are: the work published by Khaled et al. [18], which analyzes the productivity of three modeling tools (IBM Rational Software Architect (RSA), MagicDraw and Papyrus); the work published by Agner et al. [2], which determines which strengths and weaknesses of modeling tools are most important to the students; and the work published by Planas et al. [15], which analyzes the usability of two modeling tools (MagicDraw and Papyrus) from video recordings of students using the tools.

Focusing on OCL, Burgueño et al. [4], [5] describe the main issues they found when teaching modeling in a dedicated Software Engineering course, and present a simple case study that the authors developed and successfully used in class. However, in this course, all the students used the USE tool (both for modeling and for specifying the OCL constraints) and the advantages/disadvantages of using a tool, in comparison to not using any tool, were not studied and reported. On the other hand, Maraee et al. [11] compare the efficiency of developing constraints using two different approaches: the declarative OCL language using the USE tool, and the imperative Java language using a Java IDE.

To a lesser extent, there are also some studies that explicitly compare the use of tools versus writing with pen and paper

and its impact on student learning. A good part of these studies, which are mainly focused on learning programming languages, determine the advantages of electronic evaluation over traditional evaluation (on paper or oral). Bessedsen et al. [3] report an experiment with laboratory tests in an introductory programming course with more than 500 students. The result was satisfactory in all aspects: simplicity, efficiency, performance and satisfaction of the participants (both teachers and students). On the other hand, Rytkönen et al. [17] describe their experience teaching C programming. Their findings determine that students perceive programming exams in electronic format as more realistic and natural compared to exams on paper. Likewise, Bottcher et al. [6] report on their transition from paper to electronic exams for an introductory course in Java programming.

Finally, a study closer to our experience is the work published by Öqvist et al. [14], where the authors describe their experiment to evaluate the effects on student performance by comparing hand coding versus the use of programming tools. In this case, the subjects of the experiment were students of initial courses and the selected language was Java. Unfortunately, the results were inconclusive.

To the best of our knowledge, there are no similar studies focusing on the use of modeling tools versus manual drawings for graphical languages like UML or rule definition languages like OCL.

VIII. Conclusions

In this paper, we have presented a study that compares the impact of using of a modeling tool to specify OCL constraints versus the definition of the constraints on paper in a Requirements Engineering course. Our study uses different empirical methods (e.g., interviews and questionnaires) of different natures (quantitative and qualitative) and involves several subjects (students and teaching assistants) to cross-validate and confirm the validity of the results.

The study concludes that students have a positive perception towards the use of modeling tools to specify OCL constraints. Besides, teaching assistants also approve and advocate the use of such tools in subsequent editions of the course. Furthermore, the results seem to indicate that the use of MagicDraw has to some extent a positive impact on the students' final grades, which are slightly higher when they use the tool.

This study is a first attempt towards improving the learning of formal and standard languages, such as OCL, by undergraduate students as well as their satisfaction with the learning process. After this experiment, we have chosen to maintain the optional use of MagicDraw in our Requirements Engineering course and we plan to continue analyzing the evolution of our students' achievements and satisfaction.

In the future, we plan to study whether other instruments (such as the learning resources provided or other modeling tools) can help to improve the performance and satisfaction of the students. We also plan to measure how the usability of such tools can impact the perception of students and their learning experience. We will also explore solutions to be able to relate the student questionnaires with the grade obtained in the CATs at the same time that we ensure their anonymity. In the long term, the success in the use of these tools will allow us as professors to set more ambitious learning objectives both in this course and in other software engineering courses in our university.

References

[1] Jony Adkins and Diana Linville. Testing frequency in an introductory computer programming course. *Information Systems Education Journal*, 15:22–28, 2017.

[2] Luciane Telinski Wiedermann Agner, Timothy C. Lethbridge, and Inali Wisniewski Soares. Student experience with software modeling tools. *Software & Systems Modeling*, 18(5):3025–3047, 2019.

[3] Jens Bennedsen and Michael E. Caspersen. Assessing process and product – a practical lab exam for an introductory programming course. In *Proc. of Frontiers in Education (FIE)*, 2007.

[4] Loli Burgueño, Antonio Vallecillo, and Martin Gogolla. Teaching model views with UML and OCL. In *Proc. of MODELS 2017 Satellite Events*, volume 2019 of *CEUR Workshop Proceedings*, pages 533–538, 2017.

[5] Loli Burgueño, Antonio Vallecillo, and Martin Gogolla. Teaching UML and OCL models and their validation to software engineering students: an experience report. *Computer Science Education*, 28(1):23–41, 2018.

[6] Alex Böttcher, Veronika Thurner, and Daniela Zehetmeier. Alignment of teaching and electronic exams and empirical classification of errors for an introductory programming class. In *Proc. of CSEET'20*, pages 1–10, 2020.

[7] Steve Easterbrook, Janice Singer, Margaret-Anne D. Storey, and Daniela E. Damian. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering*, pages 285–311. 2008.

[8] Martin Gogolla, Fabian Büttner, and Jordi Cabot. Initiating a benchmark for UML and OCL analysis tools. In Margus Veanes and Luca Viganò, editors, *Tests and Proofs - 7th International Conference, TAP@STAF 2013, Budapest, Hungary, June 16-20, 2013. Proceedings*, volume 7942 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2013.

[9] Martin Gogolla, Fabian Büttner, and Mark Richters. USE: A UML-based specification environment for validating UML and OCL. *Science of Computer Programming*, 69:27–34, 2007.

[10] Loli Burgueño, Javier Luiz Cánovas Izquierdo, Elena Planas. MagicDraw/OCL materials (fall 2020). http://hdl.handle.net/20.500.12004/1/C/MODELSEDUSYMP/2021/001.

[11] Azzam Maraee, Eliran Nachmani, and Arnon Sturm. Constraints specification via tool support: A controlled experiment. *J. Object Technol.*, 19(3):3:1–18, 2020.

[12] Object Management Group. *Object Constraint Language (OCL) Specification. Version 2.4*, 2014.

[13] Object Management Group. *Unified Modeling Language (UML) Specification. Version 2.5*, March 2015.

[14] Martina Öqvist and J. Nouri. Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming. *Journal of Computers in Education*, 5:199–219, 2018.

[15] Elena Planas and Jordi Cabot. How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus. *Comput. Stand. Interfaces*, 67, 2020.

[16] Dassault Systemes (previously NoMagic). Magicdraw. https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/, 2021.

[17] Anni Rytkönen and Venla Virtakoivu. Comparative student experiences on electronic examining in a programming course - case C. In *Proc. of Koli Calling*, 2019.

[18] Safdar Aqeel Safdar, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. Empirical evaluation of UML modeling tools-a controlled experiment. In *Proc. of ECMFA'15*, pages 33–44, 2015.

[19] Michael de Raadt Anne Philpott Judy Sheard Mikko-Jussi Laakso Daryl D'Souza James Skene Angela Carbone Tony Clear Raymond Lister Geoff Warburton Simon, Donald Chinn. Introductory programming: Examining the exams. In *Proc. of ACE'12*, volume 123, page 61–70, 2012.

[20] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proc. of SIGCSE '11*, page 93–98, 2011.