Computer Standards & Interfaces 67 (2020) 103363

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus

Elena Planas[*,a], Jordi Cabot[b,a]

[a] Universitat Oberta de Catalunya (UOC), Barcelona Spain
[b] ICREA, Barcelona, Spain

ARTICLE INFO

Keywords:
UML
Class diagram
Papyrus
Magicdraw
Controlled experiment
GOMS

ABSTRACT

Software modeling is a key activity in software development, especially when following any kind of Model Driven Software Engineering (MDSE) process. In this context, standard modeling languages, like the Unified Modeling Language (UML), and tools for supporting the modeling activities become essential.

The aim of this study is to analyze how modelers build UML models and how good modeling tools are in supporting this task. Our goal is to draw some useful lessons that help to improve the (UML) modeling process both by recommending changes on the tools themselves and on how UML is taught so that theory and practice of UML modeling are better aligned.

Our study employs two research approaches. The main one is an empirical experiment (which analyzes screen recordings registered by undergraduate students during the construction of a UML class diagram). An analytical analysis complements the previous experiment. The study focuses on the most frequent type of UML diagram, the class diagram, and on two tools widely used by the modeling community: MagicDraw and Papyrus.

1. Introduction

Software modeling is a key activity in Model Driven Software Engineering (MDSE) [1] to increase efficiency and effectiveness in software development, as demonstrated by various quantitative and qualitative studies [2]. In this context, standard languages and tools for supporting modeling become essential. On the language side, the Unified Modeling Language (UML)[1] is the *de facto* standard for modeling software systems. UML [3] provides several diagrams for modeling the structure of a software system, its architecture and its behavior. A large number of commercial and open-source tools are available to support UML modeling such as MagicDraw, Papyrus, ArgoUML, Modelio, StarUML, among many others[2]. While the tool features differ, they all offer a graphical editor to facilitate the definition of UML models.

A large number of studies have been conducted in the literature to analyze how UML is used and how usable are UML modeling tools [4–17]. Our study adds a new perspective to these previous works by analyzing screen recordings of modelers[3] actually trying to build UML models. We focus on the most frequently used UML diagram, the class diagram [18], and on two tools widely used by the community: MagicDraw[4] (version 18.0 Personal Edition, from 2017) and Papyrus[5] (Neon 2.0.X version, from 2017). We have chosen these tools because they are two of the most popular modeling tools and they cover a representative group of modelers. MagicDraw is one of the major commercial modeling tools, recently acquired by Dassault Systems to strengthen its set of System Engineering tools while Papyrus is the reference open source modeling tool in the Eclipse community.

Many of the previous studies relied on surveys to get the raw data that limits the insights that can be derived for a usability study. We believe our real-time observations of the study participants during the modeling process itself bring new dimensions to be exploited. In this study we analyze several perspectives: the strategy followed to build the models, the effort required to build them and the obstacles arisen throughout all the modeling process. Note that our focus is to assess the usability of UML modeling tools and not to study the variations in the quality and creativity of the models defined by each participant. To

[*] Corresponding author at: Rambla del Poblenou 156, 08018 Barcelona, Spain.
E-mail addresses: eplanash@uoc.edu (E. Planas), jordi.cabot@icrea.cat (J. Cabot).
[1] UML specification: https://www.omg.org/spec/UML/.
[2] See a curated list of UML tools here: https://modeling-languages.com/uml-tools/.
[3] In this paper we use the word *modeler* to refer to all the users that use modeling tools, including students and professional developers.
[4] www.nomagic.com/products/magicdraw.
[5] www.eclipse.org/papyrus.

https://doi.org/10.1016/j.csi.2019.103363
Received 10 October 2018; Received in revised form 24 June 2019; Accepted 25 June 2019
Available online 29 June 2019

isolate as much as possible the study of this usability aspect from other dimensions, participants were asked to draw with the tool a UML model already given in advance. Therefore, in this paper, we use the term "modeling" strictly in the sense of recording a UML class diagram in a given modeling tool. This approach allows us to compare the results of our controlled experiment.

The empirical data reported in this paper is supported by more than 12 h of screen recordings (interaction with the modeling tools including verbal audio comments) registered by undergraduate students during the construction of a UML class diagram. To complement our controlled experiment, we conducted a GOMS (Goals, Operators, Methods and Selection Rules) study in order to assess the usability of the studied tools from an analytical view.

According to the objectives template of the Goal/Question/Metric method (GQM) [19], the general objective of our study is: ***Analyze*** *how students model (what strategy they follow, what effort is needed and what obstacles they encounter) and how modeling tools support this task* ***in order to*** *improve the performance of UML modeling tools as well as the way of how UML modeling is taught* ***with respect to*** *the construction of UML class diagrams* ***from the perspective of the*** *modeler*.

The rest of the paper is organized as follows: Section 2 discusses the state of the art; Sections 3 and 4 describe in more detail the perspectives of our study and its goals; Sections 5 and 6 describe our empirical and analytical studies respectively; Section 7 discuses the results of both experiments and proposes several recommendations; and finally Section 8 draws the conclusions and outlines the further work.

2. State of the art

In the last years, an increasing amount of research in software engineering has been devoted to analyze and experiment how software engineering methods and tools are being used by students and practitioners. In this section we review the most relevant related works covering software modeling aspects according to several perspectives.

The related works may be classified into two categories based on whether their focus is on the usability of the language or on the tools supporting it. Many works belong to the former [7–9,13–15,17,20,21]. These works aim to examine how UML is adopted, which are the most/less used UML diagrams, what are the main difficulties when modeling, etc. Other works look more closely to the usability of UML modeling tools themselves [4–6,10–12,16] mainly to compare the different tools among them. These are the works closer to ours.

Table 1 summarizes the studies of this second category according to its goal, the tool/s under study, the employed research approach, the size of the sample (if applicable) and the data analyzed during the study. The last row shows our own study for comparison purposes.

From Table 1 we can observe that most works employ empirical research approaches, relying on case studies [5,10–12], controlled experiments [6,16] and web-based surveys [4] to conduct the study. To a lesser extent, some works employ also analytical research approaches [6].

To the best of our knowledge, our study is the first one in this context analyzing video recordings as part of an empirical study.

Finally, although they are out of the scope of this work, we would like to briefly mention two works which have inspired our work. Aiko et al. [22] and Murphy et al. [23] analyze how Java programming tools (in particular, the IDE Eclipse) are used by practitioners through analyzing video recordings and interaction traces. As our work, these procedures allow to extract highly relevant information, through the observation in real time of all intermediate states during the programming process, instead of analyzing only the final state once the program has been constructed.

3. Perspectives of our study

As introduced before, the overall goal of this study is to analyze how

Table 1
Related works focused on analyze the usability of UML modeling tools.

Ref (year)	Goal	Tool/s	Research Approach	Sample	Data analyzed
[4] (2017)	Analyze the use of tools in modeling teaching	n/a	Empirical: Web-based survey	150 professors who taught modeling in 30 countries	Data collected from the survey
[5] (2007)	Compare the usability of two UML tools with respect to their suitability for explorative UML sketching	Rational Rose and UMLet	Empirical: Case Study	n/a	Usability data (mouse clicks) collected from 16 testing scenarios
[6] (2005)	Test empirically performance of several UML modeling tools	Visual Paradigm for UML, Enterprise Architect, Jude Community, Meta Mill, Poseidon for UML and ArgoUML	Empirical: Controlled experiment + Analytical: GOMS	58 third-year students of Computer Science course at Gdansk University of Technology	Data manually collected by the participants during the realization of the experiment's tasks
[10] (2009)	Propose a feature-based evaluation approach to determine the specification compliance of UML tools	68+ modeling tools	Empirical: Case study (tool evaluation)	n/a	Usability data collected during the case study
[11] (2011)	Compare several modeling tools, showing advantages and disadvantages for each one	UMLet, Visual Paradigm, Rational Rose, MagicDraw, ArgoUML, Enterprise Architect	Empirical: Case study	n/a	Usability data collected during the case study
[12] (2009)	Define criteria to select the best tool for building software systems	Rational Rose, ArgoUML, MagicDraw, Enterprise Architect	Empirical: Case study	n/a	Usability data collected during the case study
[16] (2015)	Compare the productivity of the software engineers while modeling with several tools	IBM Rational Software Architect (RSA), MagicDraw and Papyrus	Empirical: Controlled experiment	30 students at National University of Computer and Emerging Sciences (Islamabad, Pakistan)	Data collected from a survey conducted as part of the experiment
Our work	Analyze how two UML modeling tools (MagicDraw and Papyrus) are used	MagicDraw and Papyrus	Empirical: Controlled experiment + Analytical: GOMS	45 students at Universitat Oberta de Catalunya (Barcelona, Spain)	Screen-cast videos captured during the tools usage

UML class diagrams are built using two modeling tools (MagicDraw and Papyrus) and how well modeling tools support this task. To this end, we pay attention to three perspectives: the *modeling strategy* (see Section 3.1), the *modeling effort* (see Section 3.2), and the *modeling obstacles* (see Section 3.3).

This section introduces and motivates each one of these perspectives.

3.1. Modeling strategy

We define the *modeling strategy* as the approach employed to model the class diagram, i.e. the sequence of actions that are executed by the modeler to draw it. According to the focus and the granularity of the analysis, the modeling strategy can be decomposed into two sub-perspectives: (1) the *global modeling strategy*, i.e. the general approach followed to create the whole class diagram; and (2) the *specific modeling strategy*, i.e. the particular approach followed to create specific parts of the class diagram such as the attributes and associations.

Regarding the global modeling strategy, we define in this paper two global modeling strategies inspired in two graph theory algorithms [24]:

- *Breadth Modeling* (see Fig. 1 - left), inspired in the *Breadth First Search* graph algorithm, BFS. We consider that a class diagram is modeled following a breadth strategy when their elements are created in an orderly way by its typology, being the most usual order: first create the classes, then the attributes and, finally, the associations and the rest of the elements (associative classes, data types, etc).
- *Depth Modeling* (see Fig. 1 - right), inspired in the *Depth First Search* graph algorithm, DFS. We consider that a class diagram is modeled following a depth strategy when, the designer starts from an element (usually a class) and expands as far as possible every related element before backtracking, being the most usual order: on the basis of a class, first incorporate their attributes, then its relations with other classes, and so on.

3.2. Modeling effort

We define the *modeling effort* as the *"physical or mental activity needed to achieve something"*[6], in this case, modeling a class diagram. In our study, the modeling effort is evaluated in terms of *time* and *number of clicks* needed to model the class diagram. According to the focus of the analysis, the modeling effort can be studied from two points of view: (1) the effort devoted to initialization tasks, i.e. the work devoted to initialize the modeling tool and create an empty class diagram; and (2) the total modeling effort, i.e. the complete effort devoted to modeling the whole class diagram, including the effort devoted to initialization tasks.

3.3. Modeling obstacles

Finally, we consider the *modeling obstacles* as *"something that blocks you so that movement, going forward, or action is prevented or made more difficult"*[7] during the modeling process. In this study, we consider as obstacles the *difficulties* participants encountered with the use of modeling tools and the *mistakes* made by the modelers while using these tools. In both cases, obstacles have been analyzed and classified under the CRUD (Create, Read, Update and Delete) perspective, that is, the obstacles are contextualized according to whether they have arisen when the elements of the diagram are Created (C), Read (R), Updated (U) or Deleted (D).

[6] https://dictionary.cambridge.org/dictionary/english/effort.
[7] https://dictionary.cambridge.org/dictionary/english/obstacle.

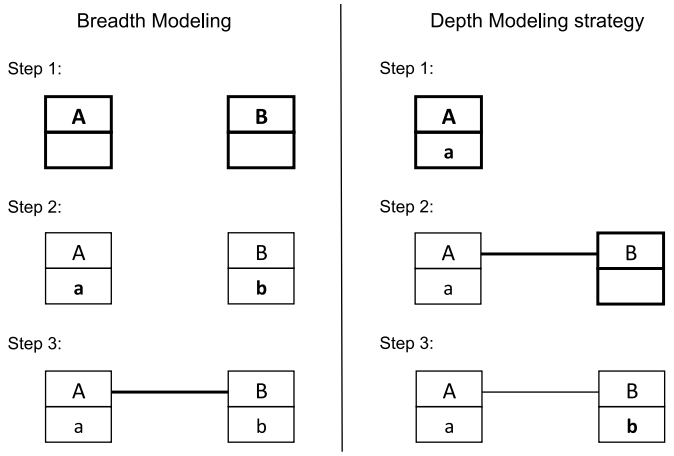

Fig. 1. Breadth Modeling (left) and Depth Modeling (right) strategies where the new elements on each step are highlighted.

4. Research questions

As introduced before, the overall goal of this study is to analyze how UML class diagrams are built and how well modeling tools support modelers on this task. In order to study this general goal, we define three research questions:

1. RQ_1: ***What modeling strategy do the modelers follow to specify UML class diagrams?***
   This RQ is centered on the first perspective of our study, the modeling strategy (see Section 3.1). In other words, it focuses on the method to built UML class diagrams, i.e. the procedure or process for attaining it.
2. RQ_2: ***How much modeling effort (time and number of clicks) is needed to model UML class diagrams?***
   This RQ is centered on the second perspective of our study, the modeling effort (see Section 3.2). In other words, it focuses on evaluating the degree to which modeling tools support and facilitate the construction of UML class diagrams through its graphical interface.
3. RQ_3: ***What are the most common obstacles (difficulties and errors) encountered by modelers when modeling UML class diagrams?***
   This last RQ is centered on the third perspective of our study, the modeling obstacles (see Section 3.3).

To answer these research questions, we carried out two studies employing different research approaches. First, an empirical study based on a controlled experiment was performed to investigate all the three above research questions. Then, an analytical study based on GOMS (Goals, Operators, Methods and Selection Rules) was performed to analyze RQ_2 from another viewpoint. The design and the results of both studies are detailed in Sections 5 and 6 respectively.

Table 2 summarizes the outlined research questions, the perspective each one is focused and the research approaches employed in the rest of this paper to address them.

5. Controlled experiment

The aim of this experiment is to empirically evaluate the three perspectives of our study described in Section 3.

In the following subsections we give an overview of the experiment (see Section 5.1) and detail the experiment design (see Section 5.2), the data collection procedure (see Section 5.3) and the results of the experiment (see Section 5.4). The interpretation of these results is provided in Section 7.

Table 2
Research questions, their perspectives and research approaches used to address them.

Research question	Perspective of study	Employed Research Approach	
		Empirical	Analytical
RQ_1	Modeling process	✓	
RQ_2	Modeling effort	✓	✓
RQ_3	Modeling obstacles	✓	

Table 3
Effort sub-variables.

	Initialization tasks	Complete modeling process
Time	Time devoted to initialization tasks	Total time for modeling
Clicks	Number of clicks devoted to initialization tasks	Total number of clicks for modeling

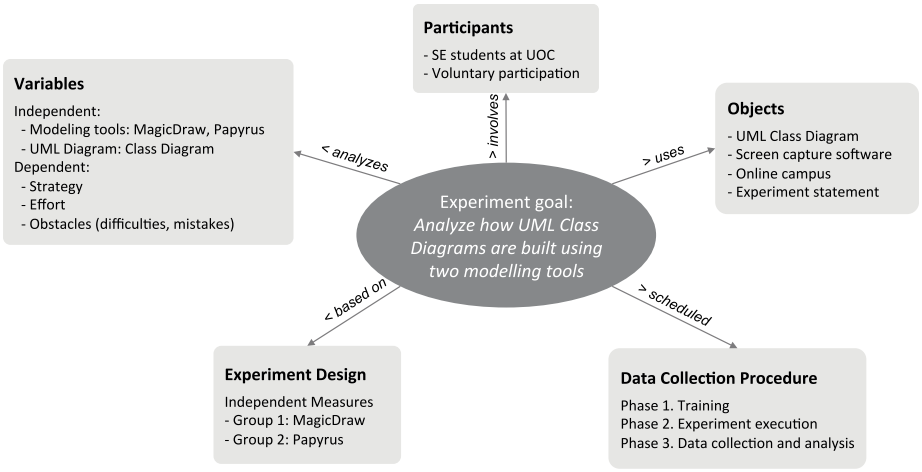

Fig. 2. Experimental design.

5.1. Overview of the overall experimental design and execution

Fig. 2 summarizes the experimental design in a conceptual map.

5.1.1. Variables

Two types of variables (independent and dependent) are involved in our experiment.

The **independent variables** (aka. factors or predictor variables) of our study are:

1. **Modeling tools**: We focus on two of the most popular modeling tools: MagicDraw (version 18.0 Personal Edition, from 2017) and Papyrus (Neon 2.0.X version, from 2017).
2. **UML diagram**: We select the most widely used UML diagram [18], that is, the UML class diagram.

The **dependent variables** (aka. response variables) of our study are:

1. **Strategy**: As introduced before (see Section 3.1) it refers to the approach used to model the class diagram.
2. **Effort**: As introduced before (see Section 3.2) it refers to the work devoted to model the class diagram. This variable is specialized in the sub-variables detailed in Table 3.
3. **Obstacles**: As introduced before (see Section 3.3) obstacles are the *difficulties* with the use of modeling tools and the *mistakes* made by the participants during the modeling process.

5.1.2. Objects

The objects used during the experiment are:

- UML Class Diagram: A UML class diagram (see Fig. 3) was provided as an object for the experiment. It consists of three classes, an associative class, a data type, two associations, a generalization, and eight attributes. The given class diagram was modeled using both MagicDraw and Papyrus.
- Screen capture software: Students were encouraged to use one of the following available screen capture software to record their videos: Screencast-O-Matic[8] (free, online, available for windows/mac), Recordmydesktop[9] (open source, linux), Camstudio[10] (free, available for windows) or Camtasia[11] (trial license, available for windows/mac), although they could choose any other software.
- Online campus of the university: The UOC online campus[12], where all the courses are taught, was the platform used to send the individual participation invitations.
- Experiment statement: The experiment statement provided the instructions to participate in the experiment. These instructions included three actions: (1) to model the provided UML class diagram using the assigned tool (MagicDraw/Papyrus); (2) in parallel, to record the entire modeling process using the screen capture software; and (3) finally, to send the recorded video through the online campus of the university. Besides, at the beginning of the video, students were asked to provide information about their previous experience with modeling and modeling tools. Also, they were encouraged to verbalize their impressions, doubts and difficulties during all the modeling process. The experiment statement was exactly the same for both groups.

5.1.3. Participants

The target participants of our experiment are undergraduate students enrolled in a *Software Engineering* course at the Universitat Oberta de Catalunya (UOC), an online university. This course is a mandatory subject within the Computer Science degree, where modeling is introduced for the first time. At the beginning of the course, students receive a list of suggested modeling tools they can use, although

[8] www.screencast-o-matic.com.
[9] www.recordmydesktop.sourceforge.net.
[10] http://camstudio.org/.
[11] http://discover.techsmith.com/camtasia-brand-desktop.
[12] www.uoc.edu.

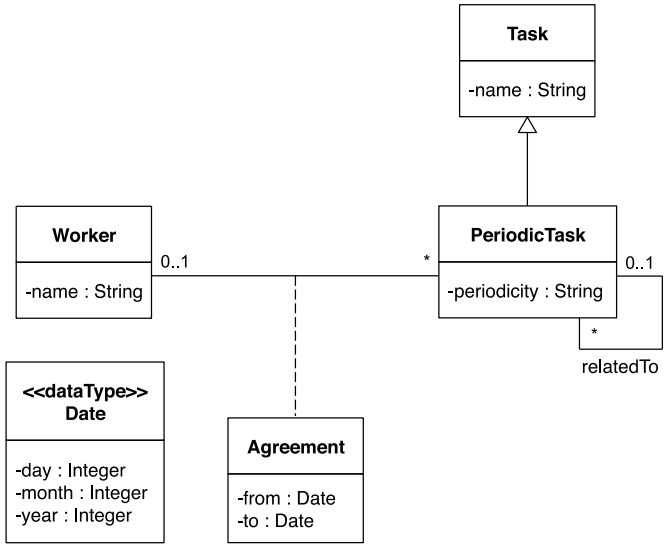

Fig. 3. UML Class Diagram of our experiment.

students can freely choose the tool they will use.

The participation of the students in the experiment was voluntary, that is, although all the students enrolled in the course were invited to participate, only a subset of them chose to participate. To motivate their participation, all participants were rewarded with an extra score at the final grade of the course.

5.1.4. Sample

The experiment was conducted under the context of the TeSLA project[13], a project funded by the European Commission and coordinated by the Open University of Catalonia (UOC). Roughly, the TeSLA project provides to educational institutions, an adaptive trust e-assessment system for assuring e-assessment processes in online and blended environments. It aims to support both continuous and final assessment to improve the trust level across students, teachers and institutions. In 2017, the *Software Engineering* course participated in a pilot of the TeSLA project, where all the enrolled students could chose voluntary to participate. All participants signed an informed consent to agree the data collected during the pilot and their uses.

A total of 202 students were enrolled in the course. Although a total of 65 students (32% of the enrolled students) were initially interested to participate in our experiment, the participation in the study was finally comprised for a total of 45 students (i.e. 20 students drop out) - 4 women (9%) and 41 men (91%) -, representing the 22% of the total students enrolled in the course. The expertise level of the participants with the tools before executing the experiment was captured. As shown in Fig. 4, the experience of the participants using both modeling tools, according to her own point of view, was low, although the students which used MagicDraw reported a little more expertise wrt the students which used Papyrus.

5.2. Experiment design

The experimental design applied to conduct this study is *independent measures*. An independent measures design assigns participants to separate groups. Each of the groups is then designated to a single research condition.

In our experiment, the participants were allocated into two experimental groups, according to the group to which they were enrolled. This explains the different size of both groups (the first group had 27 participants, while the second group had 18 participants). A different modeling tool was assigned to each group to conduct the experiment: the first group used MagicDraw and the second group used Papyrus. The participants did not know their assigned tool in advance.

The experiment was conducted at the end of the semester, so the students were trained about modeling UML class diagrams before participating at the experiment.

The experiment was limited to 15 min, so if the students had not been able to model the diagram at that time, they had to abort the process. In order to process the data, we fix a total time of 15 min to those students who did not complete the task. We do not omit these observations since they are interesting regarding the modeling strategy and specially the modeling obstacles.

5.3. Data collection procedure

The experiment was planned in a term with the following schedule:

- Weeks 1 to 11: **Training**. Before conducting the experiment, the participants, together with the rest of their classmates, were trained about modeling with UML. This phase includes training on the construction of UML class diagrams with the tool chosen by the students (which was not necessarily the tool used during the experiment).
- Weeks 11 to 14: **Experiment execution**. The experiment was released at the 11st week of the semester and the students had exactly three weeks to run the experiment, i.e. to download the instructions from the online campus, to learn the assigned tool, to record their videos and deliver them through the online campus again.
- After week 14: **Data Collection and Analysis**. Finally, all the videos were downloaded and manually analyzed by the researchers. For each video, we manually collected data from the three variables of the study:
  - Strategy: After the visualization of the each video, the strategy used to model the whole class diagram was classified into Breadth Modeling, Depth Modeling or alternating strategy. In a similar line, other behaviours observed during the modeling of other specific elements (like classes or attributes) were manually collected.
  - Effort: The time as well as the number of clicks devoted to model the whole class diagram was manually computed during the visualization of each video.
  - Obstacles: The obstacles (difficulties and errors) encountered by modelers during the experiment were manually collected during the visualization of each video.

    To analyze each video, it was (re)reproduced and paused repeatedly to carefully collect all the relevant information. The videos were analyzed primarily by one of the researchers. Anyway, in order to validate our study, 18% of them were also analyzed by both researchers.

5.4. Results

This section summarizes the results of the collected data from our experiment regarding the three study perspectives: the modeling strategy (Section 5.4.1), the modeling effort (Section 5.4.2) and the modeling obstacles (Section 5.4.3). These results are analyzed and interpreted in Section 7 together with the results of the analytical experiment.

A total of 45 observations, more than 12 h of screen recordings, were analyzed, 27 (60%) of which used MagicDraw and 18 (40%) of which used Papyrus.

5.4.1. Strategy

As introduced before, the *strategy* variable refers to the approach used to model the class diagram.

Global modeling strategy

[13] www.tesla-project.eu/partner/uoc.

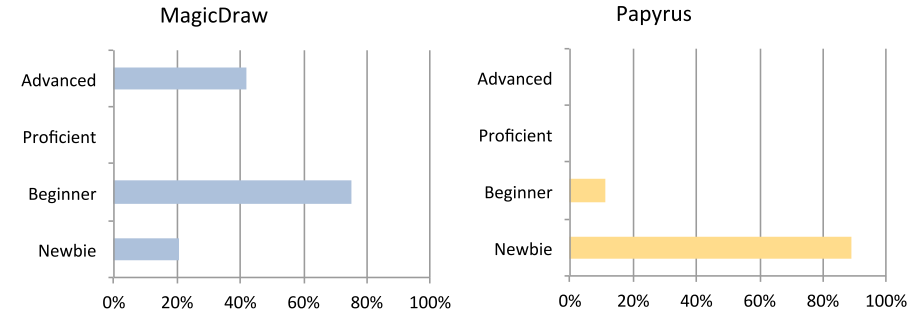

Fig. 4. Expertise level of the participants with MagicDraw and Papyrus.

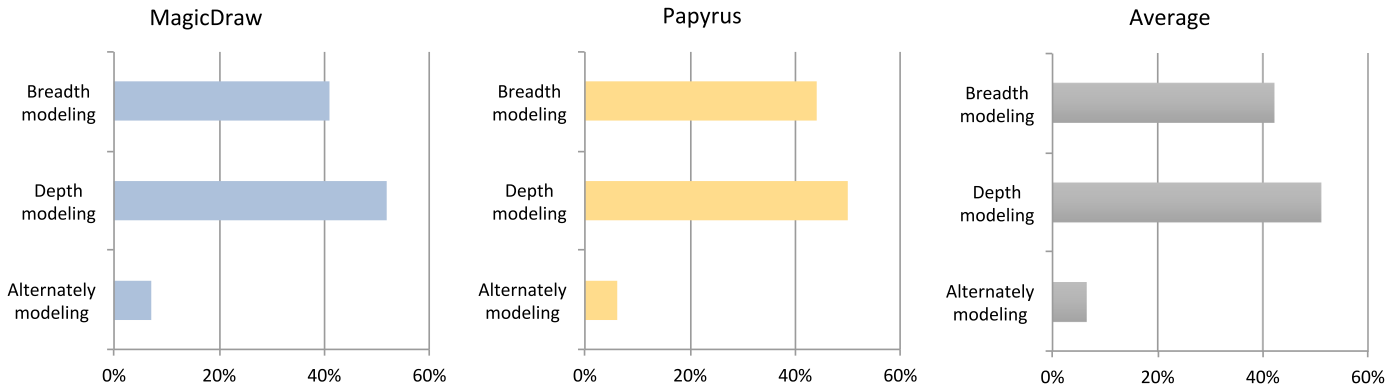

Fig. 5. Modeling strategies followed by the participants during the construction of the whole class diagram.

Fig. 5 reports the global modeling strategy during the construction of the whole class diagram. The chart on the left (blue) shows the data of the first group (MagicDraw), the chart on the middle (yellow) shows the data of the second group (Papyrus) and the chart on the right (grey) shows the average between both tools. As can be seen in Fig. 5, the participants followed the two previously introduced strategies (breadth and depth modeling) in a balanced way in both groups: an average of 51% of the participants (52% in the 1st group (MagicDraw) and 50% in the 2nd group (Papyrus)) used the breadth modeling strategy, in front of an average of 42% of the participants (41% in the 1st group (MagicDraw) and 44% in the 2nd group (Papyrus)) that used a depth modeling strategy. To a lesser extent, an average of 7% of the participants (7% in the 1st group (MagicDraw) and 6% in the 2nd group (Papyrus)) modeled the diagram following an alternating strategy. As can be seen, in average, there is no relevant difference in terms of global modeling strategy depending on the modeling tool.

On the other hand, we want to highlight that in 93% of the cases the first element modeled was a class, whereas in only 7% of the cases the first was the data type *Date*.

Specific modeling strategy

Regarding the specific modeling strategy involving the creation of attributes and associations, we analyze the specific order in which their properties (name, multiplicity, visibility, etc.) were updated.

During the creation of the attributes, in most of the cases in both groups, the first specified property was its name, followed by its type and, only in some cases, its visibility and multiplicity in random order. As can be seen in Fig. 6, 98% of the participants (96% in MagicDraw and 100% in Papyrus) indicated the name of the attributes and 80% (85% in MagicDraw and 72% in Papyrus) also indicated its type. On the other hand, only 22% of the participants (7% in MagicDraw and 44% in Papyrus) indicated the visibility of the attributes and 20% (11% in MagicDraw and 33% in Papyrus) indicated its multiplicity. In all the cases, the most usual order to modify the properties of the attributes was: name, type and visibility/multiplicity (these last two alternately).

During the creation of the associations, in most of the cases in both groups, the first specified property was its multiplicity, followed by its name and, only in a few cases, its navigability and the role names in random order. As can be seen in Fig. 7, 87% of the participants (100% in MagicDraw and 67% in Papyrus) indicated the name of one or more associations and 82% (81% in MagicDraw and 83% in Papyrus) also indicated its multiplicity. Other properties such as navigability and the role names of associations were only modified by the participants of group 2 (Papyrus), although they only represent 28% (navigability) and 11% (role names) respectively.

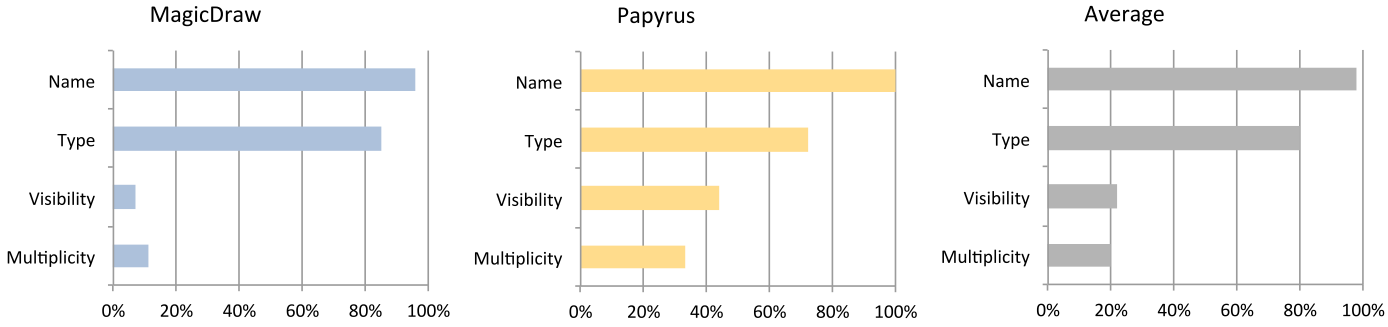

Fig. 6. Initialized properties during attribute's modeling.

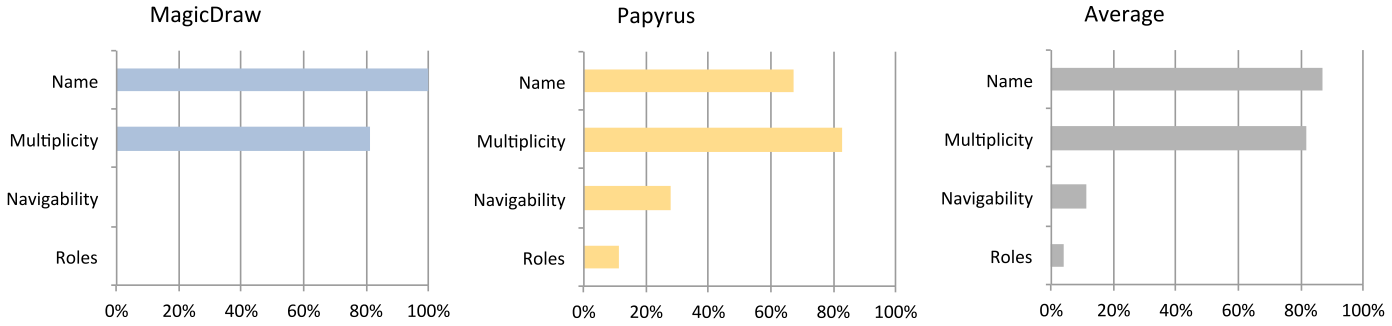

Fig. 7. Initialized properties during association's modeling.

In addition to the above, we observed some other common behaviours related with the strategy. For instance, only one student (from the second group) saved the project just after create the empty project, while most of the students did not save the project throughout the experiment.

5.4.2. *Effort*

As introduced before, the *effort* variable represents the work devoted to model the class diagram.

As the time to construct the whole class diagram was limited to 15 min, not all participants finished the proposed task. In the first group (MagicDraw) 1 participant (4%) did not finish the whole class diagram; while in the second group (Papyrus) 5 participants (28%) did not finish it.

Table 4 presents descriptive statistics for the *effort* variable including measures for central tendency and dispersion. As can be observed, in all the measures, the results obtained by the MagicDraw participants are better than the results obtained by the Papyrus participants.

Figs. 8 and 9 show the total effort in both groups aggregated by categories and Fig. 10 shows the dispersion of the total effort regarding both groups. These figures show a subset of the descriptive statistics shown on Table 4.

In addition to the above, we observed some other relevant behaviours related with the effort. For instance, around half of the students on each group were concerned about the aesthetics of the diagram, investing a considerable amount of time trying to align the elements of the diagram.

Moreover, we assessed whether the distributions of the variables *time* and *number of clicks* were significantly different for MagicDraw and Papyrus. This assessment is usually performed by applying the Student's *t*-test to compare two independent samples, which tests a statistically significant difference between the distributions according to some known probability density function under the null hypothesis. Student's *t*-test assumptions require the data to both follow a normal distribution and have homogeneity of variances. To check these assumptions we applied the *Saphiro-Wilk normality test* for the former and the *Barlett test* for the later. The results for both tests are shown in Table 5. As can be observed, all variables passed both the normality and homogeneity of variances tests (see rows 1–4 and 5–6, respectively). Then, we could trust the Student's *t*-test results (see rows 7–8). As the tests outputs return an overall *p*-value, to avoid clutter when reporting *p*-values, we superscript the results using the following convention: no superscript corresponds to p-value > = 0.05, * corresponds to 0.01 < = p-value <

Table 4
Descriptive statistics for the *effort* variable (time and number of clicks) considering the effort devoted to initialization tasks (until the first UML element is created) and the total effort (until the whole Class Diagram is created) using MagicDraw (MD) and Papyrus (P).

Measure type	Measure	Time (s)				Number of clicks			
		Initialization		Total		Initialization		Total	
		MD	P	MD	P	MD	P	MD	P
Central tendency	Mean	51.3	68.2	502.4	615.2	8.5	12.6	118	150.1
	Median	30	56.5	515.5	608	7	9.5	112	140.5
	Mode	30	N/A	430	480	6	11	90	133
Dispersion	Min	14	33	240	440	2	6	62	98
	Max	145	183	840	788	25	37	192	245
	Range	131	150	600	348	23	31	130	147
	Standard deviation	37.4	39.4	164.5	116.2	4.8	7.8	32.6	35

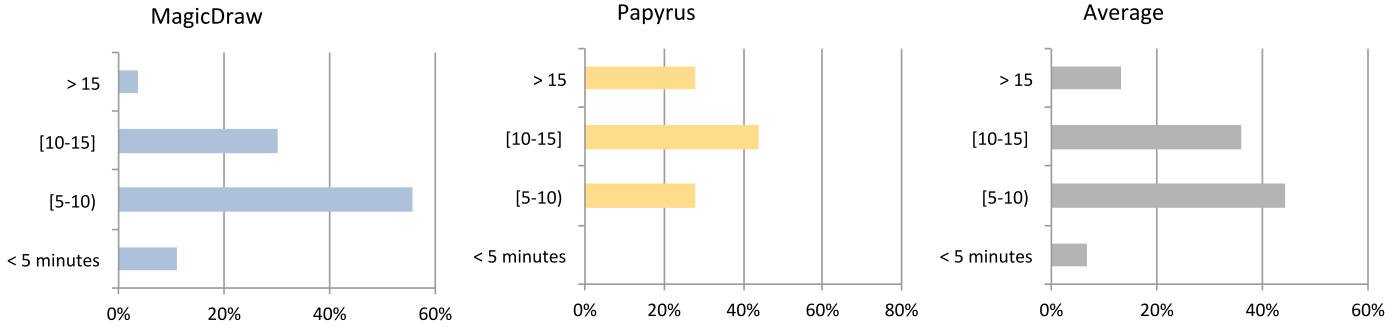

Fig. 8. Total modeling time aggregated by categories.

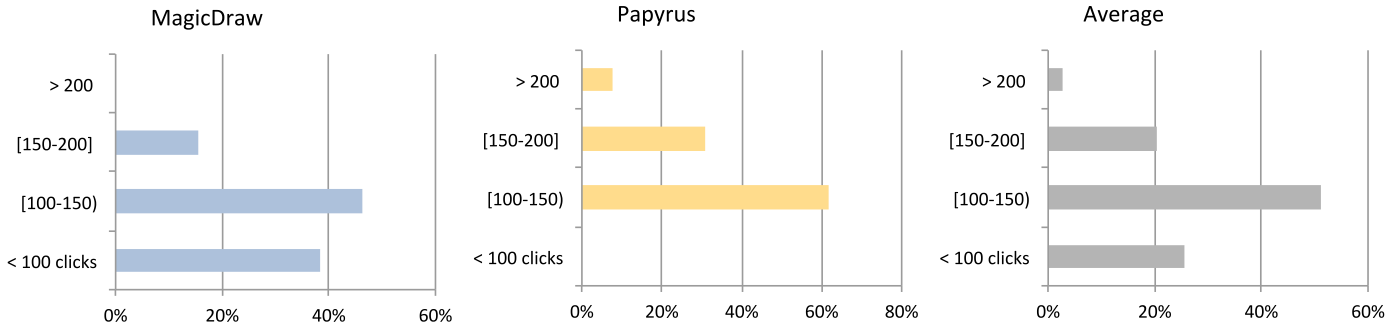

Fig. 9. Total modeling clicks aggregated by categories.

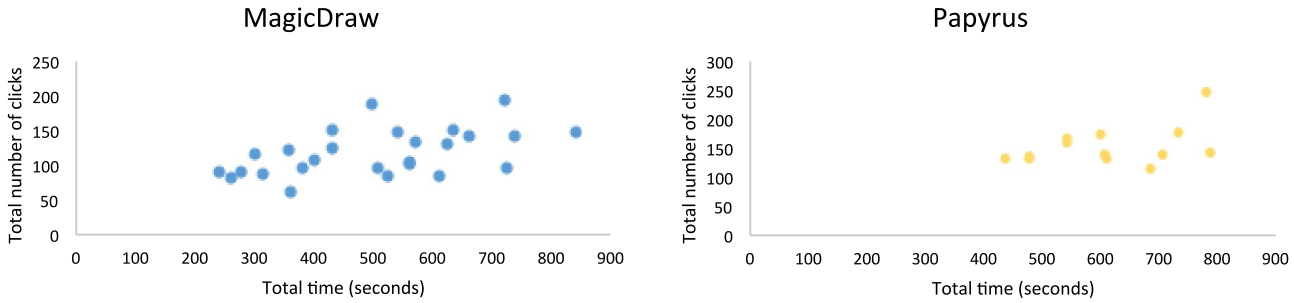

Fig. 10. Dispersion of the effort in MagicDraw and Papyrus.

Table 5
Results of the applied tests. Column *p*-value indicates whether the variable pass the corresponding test (✓value) or not (✗value). If the variable do not pass the test, we indicate the interval of the *p*-value: no superscript corresponds to *p*-value > = 0.05, * corresponds to 0.01 < = *p*-value < 0.05, ** corresponds to 0.001 < = *p*-value < 0.01 and *** corresponds to *p*-value < 0.001.

#	Test	Variable	*p*-value
1	Shapiro-Wilk normality test	Total time - MagicDraw	✓
2		Total time - Papyrus	✓
3		Total number of clicks - MagicDraw	✓
4		Total number of clicks - Papyrus	✓
5	Bartlett's test (homogeneity of variance)	Total time MagicDraw-Papyrus	✓
6		Total number of clicks MagicDraw-Papyrus	✓
7	t-Student test	Total time MagicDraw-Papyrus	✗*
8		Total number of clicks MagicDraw-Papyrus	✗**

0.05, ** corresponds to 0.001 < = *p*-value < 0.01 and *** corresponds to *p*-value < 0.001.

Our results reveal that there are significant differences in the distributions of both variables with regard to the tool, which means that the effort (measured as *time* and *number of clicks*) is significantly different when using MagicDraw and Papyrus.

5.4.3. Obstacles

As introduced before, the *obstacles* variable represents the *difficulties* and the *mistakes* made by the participants using the target modeling tools.

Difficulties

Table 6 describes the most frequent difficulties risen during the modeling process, classified according the CRUD perspective and Fig. 11 shows their frequency on each tool. Clearly, the most relevant difficulty in both environments, but especially in Papyrus, was the creation of the association class (D1). 33% of the participants in the 1st group (MagicDraw) had difficulties creating the association class. This percentage rises up to 78% in the 2nd group (Papyrus). Analyzing the videos, we observed that the participants had serious difficulties to choose the right element of the palette of elements to create associative classes (trying several elements as a *class* or an *association*). In addition, although in MagicDraw there is a unique way to create an associative class, in Papyrus there are several alternatives, which confuse and disorients the participants. On the other hand, 100% of the participants of the second group (Papyrus) who achieved the task of creating an associative class through the use of the *AssociationClass* node of the palette of elements, had later difficulties to modify the multiplicities of the association (D3), since the tool has a bug that does not allow to modify the multiplicities of an association created using that node.

Mistakes

Table 7 describes the most frequent mistakes risen during the modeling process, classified according the CRUD perspective and Fig. 12 shows their frequency on each tool. The most repeated mistake made by the participants of the 1st group (MagicDraw) was to connect a simple class with a previously created association instead of using the *AssociationClass* element (M1), with a frequency of 33%. On the contrary, this mistake did not occur in the 2nd group (Papyrus), since this tool allows associating a simple class with a previously created association to create an associative class. On the other hand, the most repeated mistake made by the participants of the 2nd group (Papyrus) was the misuse of the connectors to create an inheritance, with a frequency of 33%.

In addition to the above difficulties and mistakes, we observed some other, less frequent, obstacles. For instance, one student of the first group commented that the MagicDraw tool crashed on several occasions, having to perform the experiment again. On the other hand, one student pointed out problems during the installation of Papyrus on Mac. Besides, as can be observed in some videos, few students also had problems using the tools because of the resolution of the screen they used. Finally, as a curiosity, although the used version of Papyrus contain a bug to assign multiplicities on association classes, only one student contacted the teachers to ask how they could assign such multiplicity. As explicitly evident in most of the videos, all the above mentioned obstacles generated a considerable feeling of frustration among the students.

Table 6
Difficulties risen during the modeling process classified according the CRUD perspective.

Identifier	Type	Description
D1	Create	Create an association class
D2	Read	Visualize the properties of a DataType once created (in MagicDraw, they are not displayed by default)
D3	Update	Update the multiplicity of an association class (not possible in Papyrus)
D4	Delete	Delete the navigability, visibility and role names of an association
D5	Update	Assign a previously created DataType to an attribute
D6	Update	Update the multiplicity of associations
D7	Update	Update the name of a DataType
D8	Update	Update the properties of a previously created element (due to not having the Properties tab opened)

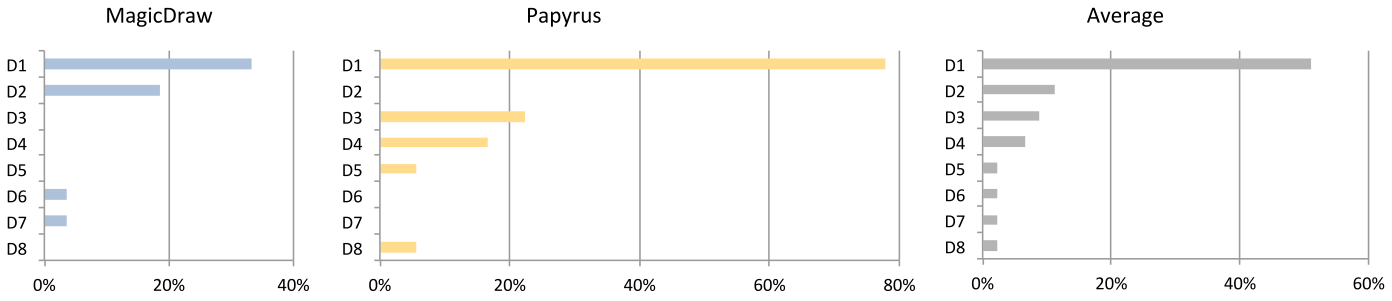

Fig. 11. Difficulties risen during the modeling process.

Table 7
Mistakes risen during the modeling process classified according the CRUD perspective.

Identifier	Type	Description
M1	Create	Connect a class with a previously created association instead of using the *AssociationClass* element
M2	Create	Create a dependency, abstraction or association instead of a generalization
M3	Create	Try to create an attribute of a data type not previously created
M4	Update	Assign the name and/or the multiplicity of an association using roles or comments
M5	Create	Create a model inconsistent with the model provided
M6	Update	Assign the multiplicities of an association in a swapped way
M7	Create	Create a class with stereotype Type instead of a DataType or create an enumeration instead of a DataType
M8	Update	Not being able to assign a basic UML type to an attribute because it has not been imported previously
M9	Create	Create a note instead of class or create *x* copies of the same class instead of creating *x* classes
M10	Create	Create a reflexive association instead of an associative class or create a generalization instead of a binary relationship
M11	Create	Create additional/redundant elements (and delete them)
M12	Create	Create an element already previously created
M13	Create	Create a Data Type instead of a Property

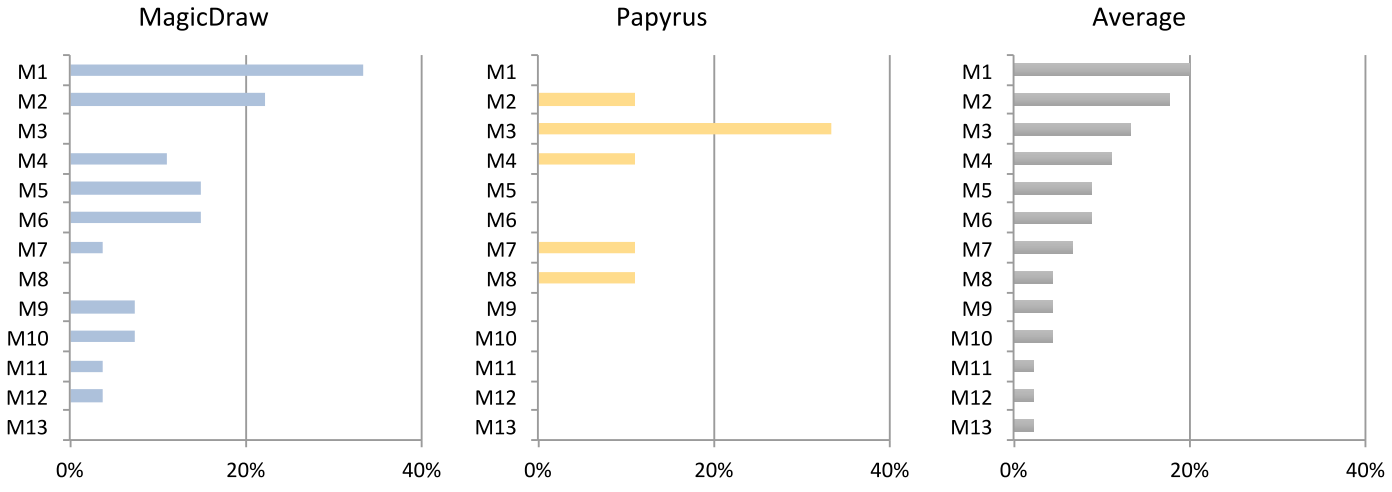

Fig. 12. Mistakes risen during the modeling process.

Table 8
Estimated execution time for operators.

Operator	Description	Estimated time (sec)
K	Pressing a key or button on the keyboard	0.12 (expert typist), 0.20 (average skilled typist), 0.28 (average non-secretarial typist), 1.2 (worst typist)
T(n)	Type a sequence of *n* characters on a keyboard	$n \times$ K
P	Point with mouse to a target on the display	1.1
B	Press or release mouse button	0.1
BB	Double-click mouse button	0.2
H	Home hands to keyboard or mouse	0.4
M	Mental act of routine thinking or perception	0.6 - 1.35; use 1.2
W (t)	Waiting for the system to respond (time t must be determined)	t

Even though this is not directly related with obstacles, some students pointed out several specific improvements regarding both tools that we want to state in this study. For instance, improving the aesthetics of MagicDraw ("given it is currently outdated" expressed by the student) or extending the tools to allow collaborative work.

5.5. Threats to validity

In this section, we discuss the main threats to validity that can affect the results of our experiment.

Our work is subjected to a number of threats to validity, namely: (1) internal validity, which is related to the inferences we made; and (2) external validity, which discusses the generalization of our findings.

Regarding the internal validity, our threats are mainly associated with the participants and the measurements. First, the participants of our experiment could have a different previous knowledge of the tools before the execution of the experiment (though we explicitly asked for this information and at least what they claimed level of expertise show a reasonable homogeneity among them). Secondly, since each student executed the experiment at their place and using their own settings, we cannot ensure identical conditions. For instance, some participants had the target diagram printed in paper while others had to visualize it on the screen (which implies to switch the screen in case they did not have multiple screens). Also, after visualizing the videos, we realized that some students had suffered from model visualization problems because of the resolution of their screens. Regarding the measurements, the data we collected could be error-prone given that the videos were manually processed. To minimize this, as commented before, an 18% of the videos were analyzed by both researchers and their results were compared, with the deviation being below 13%.

Regarding the external validity, our threats are related to our selection of the modeling tools (MagicDraw and Papyrus). Since they represent a subset of the available modeling tools, the results of this experiment should not be generalized to the global population of modeling tools even if our personal experience suggests that indeed at least some of the problems and feedback participants gave applies indeed to a large number of tools we have used ourselves

6. GOMS analysis

In order to address in more detail the RQ_2 described in Section 4, now from a more analytical perspective, this section evaluates the effort necessary to create class diagrams using MagicDraw and Papyrus by means of a well-known analytical method called GOMS (Goals, Operators, Methods and Selection Rules).

This method, beginning as a theoretical model in HCI (Human-Computer Interaction), attempts to predict and evaluate the usability of tools, interfaces or any complex applications. GOMS describes the procedures required for accomplishing a general set of tasks by decomposing the tasks into four elements [25]:

- **G**oals, i.e. something that the user tries to accomplish, often divided into subgoals.
- **O**perators, i.e. basic actions that the software allows the user to take (e.g. commands, menu selections, button clicks or direct-manipulation actions).
- **M**ethods, i.e. well-learned sequences of subgoals and operators that can accomplish a goal.
- **S**election rules, i.e. personal rules that users follow in deciding what method (if there is more than one method to accomplish the same goal) to use in a particular circumstances.

We use a simplified version of the GOMS model called Keystroke-Level Model (KLM) [26] which uses only keystroke-level operators to predict task execution time from a specified design and specific task scenario. The estimated time to accomplish each task, according to [26], is presented in Table 8. The analysis of UML modeling tools using GOMS, and KLM in particular, consisted of the following steps: (1) create a hierarchy of goals (one for both tools), (2) describe all the methods (basic action sequences) for each UML modeling tool, (3) select the operators and list their corresponding times, (4) sum the times of the operators for each method, and, finally (5) compare the results among the tools.

In the next subsections we describe in more detail the goals and methods of our KLM analysis (see Section 6.1) and the resulting time (see Section 6.2). The interpretation of the GOMS analysis is described in Section 7 together with the analysis of the controlled experiment.

6.1. Goals and methods

Following the GOMS terminology, our main goal is to create a UML class diagram for a system. This goal is decomposed into sub-goals associated with creating several parts of the class diagram: create a new project, create an empty UML class diagram, create a simple UML class with name, add attributes to a class, create an association, etc.

In the following, we exemplify the steps to accomplish several of the above goals (in particular: create a new project, create an empty class diagram, add an attribute (with name and type) to an existing class, create a binary association (with name and multiplicity) between two existing classes, and create an association class (with name and multiplicity) between two existing classes). In all cases, we assume three facts: (1) the analyzed tools are installed and opened before performing the methods; (2) at the beginning of each method the user hand starts on mouse; and (3) the users are skilled typists (55 wpm) on average. Additionally, we omit mental operators (assuming the diagram was transcribed but not created in that precise moment, as it was the case in the empirical study) and we fix the waiting time to 2 s.

The specification for the sub-goal *create a new Project* with the corresponding operators and estimated time is presented in Table 9 (MagicDraw) and Table 10 (Papyrus). The complete specification of the rest of the sub-goals can be found at https://hdl.handle.net/20.500.12004/1/J/CSI/633.

Table 9
Specification and evaluation of the method *create a new Project* using the main menu of MagicDraw.

Step	Description	Operator	Estimated Time (sec)
1	Point the "File" menu	P	1.1
2	Click the "File" menu	B	0.1
3	Point the "New Project" option	P	1.1
4	Click the "New Project" option	B	0.1
5	Point the "Name" field	P	1.1
6	Move hand from the mouse to the keyboard	H	0.4
7	Keystroke the project name (we assume 5 letters)	T(5)	1
8	Move hand from the keyboard to the mouse	H	4
9	Point the "Ok" option	P	1.1
10	Click the "Ok" option	B	0.1
11	System response time	W(2)	2
	Total estimated time (sec)		**8.5**

6.2. Results

Table 11 summarizes the results of our KLM analysis for the different sub-goals (see each row). For each tool (MagicDraw and Papyrus) we compute the number of steps and the estimated time to achieve the specific sub-goal. It is important to note that there exist several methods (i.e. sequences of actions) to perform the same task using a specific tool. For instance, in MagicDraw the modeler can create a new project from the main menu (as specified in Table 9), from the quick menu or using quick-shorts, while in Papyrus the modeler can create a new project from the main menu (as specified in Table 10) or from the quick menu. In case there is more than one method to perform the same task, Table 11 shows the average number of steps and estimated time for each tool. Finally, last column of the table shows the average of both tools.

Using the results of this GOMS analysis, we can predict the expected time to accomplish the modeling goal asked in our previous controlled experiment (see Section 5 and Fig. 3). As shown in Table 12, according to the GOMS analysis, there is an estimated difference of 20 sec regarding MagicDraw and Papyrus in order to build the same UML class diagram.

Table 10
Specification and evaluation of the method *create a new Project* using the main menu of Papyrus.

Step	Description	Operator	Estimated Time (sec)
1	Point the "File" menu	P	1.1
2	Click the "File" menu	B	0.1
3	Point the "New" sub-menu	P	1.1
4	Point the "Other" sub-menu	P	1.1
5	Click the "Other" sub-menu	B	0.1
6	Point the "Papyrus" option	P	1.1
7	Click the "Papyrus" option	B	0.1
8	Point the "Papyrus project" option	P	1.1
9	Click the "Papyrus project" option	B	0.1
10	Point the "Next" option	P	1.1
11	Click the "Next" option	B	0.1
12	Point the "Next" option again	P	1.1
13	Click the "Next" option again	B	1.1
14	Point the "Name" field	P	1.1
15	Move hand from the mouse to the keyboard	H	0.4
16	Keystroke the project name (we assume 5 letters)	T(5)	1
17	Move hand from the keyboard to the mouse	H	0.4
18	Point the "Finish" option	P	1.1
19	Click the "Finish" option	B	0.1
20	System response time	W(2)	2
	Total estimated time (sec)		**14.4**

7. Interpretation and recommendations

In this section we discuss and interpret the findings from the empirical experiment (see Section 5) as well as the analytical analysis (see Section 6), according to the research questions outlined in Section 4.

Besides, in order to overcome the detected challenges, we suggest several recommendations to improve the performance of UML modeling tools along with the way of how UML modeling is taught.

RQ_1: What modeling strategy do the modelers follow to specify UML class diagrams?

In terms of the modeling strategy, the most relevant finding from our empirical experiment is that there is no relevant difference in terms of the global modeling strategy (breadth, depth or alternately modeling). In all cases, the vast majority started by modeling the classes (93% of the time the first modeled element was a class).

This behavior is in part due to the unrestricted approach followed by the modeling tools that let modelers draw diagrams as they please. And while this freedom is, in principle beneficial, it adds a new hurdle to novel modelers that would probably prefer a more systematic approach and guidance to model.

As such, we recommend either teachers or (even better) tools to suggest a modeling strategy to novice modelers until they feel empowered enough to adapt the strategy to their own perspective on how to be most efficient at modeling.

A typical strategy to be used could be, for instance, using a top-down view (which directly derives from the breadth modeling strategy proposed in this work), which consists in: (1) first, identify and model the classes (*which classes do we need?*), (2) then, identify and model associations (*how are the classes connected?)*, and (3) finally, identify attributes and multiplicities (*what do we want to know about the objects?*). If enforced by the modeling tools, this functionality should be optional and be switched on/off by the modeler at will.

Even though we suggest recommending a specific modeling strategy, the results of our empirical experiment did not find significant differences in terms of effort (time and number of clicks) wrt the used strategy. In particular, the collected data reveals that the students who followed the depth modeling strategy took 9 min on average to model the whole class diagram, while the students who followed the breadth modeling strategy took 10 min and the students who followed an alternately strategy took 11 min. Similarly, regarding the number of clicks, the students who followed the depth modeling strategy took 127 clicks on average to model the whole class diagram, while the students who followed the breadth modeling strategy took 133 clicks and the students who followed an alternately strategy took 119 clicks.

In fact, the ideal modeling strategy, especially when it comes to the modeling of specific individual types of elements like attributes or associations, can also depend on the particular UI of the modeling tool. For instance, in MagicDraw a user can create an attribute directly introducing its name and type inside the class box (and obviate the rest of its properties unless she explicitly wants to indicate them by first opening the extended menu). Instead, in Papyrus each time a user creates an attribute, the extended menu is automatically opened at the bottom of the screen and all its properties are visualized. This could explain why only the users of the second group specified some properties such as the visibility and multiplicity of attributes.

RQ_2: How much modeling effort (time and number of clicks) is needed to model UML class diagrams?

From our empirical experiment, we observed that the effort (measured as time and number of clicks) devoted to build a class diagram is significantly different when using MagicDraw and Papyrus. This difference is related with the obstacles the participants found using both tools (slightly more in Papyrus), as we will discuss in RQ_3. As a curiosity, 13% of the participants (considering both groups) did not finish the whole class diagram in 15 min. This is clearly a trade-off to

Table 11
Summary of the number of steps and estimated time for achieving several sub-goals using MagicDraw and Papyrus.

Sub-goal	MagicDraw		Papyrus		Average	
	Number of steps	Estimated time (sec)	Number of steps	Estimated time (sec)	Number of steps	Estimated time (sec)
0. Create a new project	8	6.8	17.5	12.95	**12.75**	**9.88**
1. Create an empty Class Diagram	9.67	5.83	24	16.3	**16.83**	**11.07**
2. Create a UML class (with name)	11	6.25	9.5	5.6	**10.25**	**5.93**
3. Add an attribute (with name and type) to an existing class	13.5	8.2	15.5	9.4	**14.5**	**12.9**
4. Create a binary association (with name and multiplicity) between two existing classes	20	11.6	17	10.2	**18.5**	**10.9**
5. Create an association class (with name and multiplicity) between two existing classes	25	15.1	26	15.6	**25.5**	**15.35**

convince software developers to use modeling tools.

Moreover, from our analytical experiment we conclude that, during the initialization tasks (creating a new project and an empty UML class diagram), MagicDraw is more usable in terms of efficiency (amount of steps and estimated time). For instance, on average, MagicDraw requires 9.5 steps and 6.15 s less than Papyrus to create a new project (sub-goal 0) and 14.33 steps and 10.47 s less than Papyrus to create an empty class diagram (sub-goal 1). However, in the rest of the tasks devoted to the construction of the UML class diagram (sub-goals 2–5) both MagicDraw and Papyrus present similar usability. In particular, MagicDraw is slightly more efficient when adding attributes (sub-goal 3) and creating association classes (sub-goal 5) while Papyrus is more efficient when creating UML classes (sub-goal 2) and binary associations (sub-goal 4).

When taking into account both experiments, we conclude that globally the effort and efficiency could be improved in both tools. Many times modelers were not able to optimally use the tools and end up being much more inefficient than the optimal solution. For instance, when the tool offers several alternatives to achieve the same goal (e.g. creating an association class), modelers do not always choose the simplest path. Similar trade-off between freedom and efficiency as above. We encourage tools to rethink some of their UI components and be more opinionated with their interface in order to prevent clearly inefficient modeling strategies.

RQ_3: What are the most common obstacles (difficulties and errors) encountered by modelers when modeling UML class diagrams?

Section 5.4.3 reports on the list of modeling obstacles modelers encountered during the experiment. Many of these obstacles are due to the inexperience of the modelers. For instance, in many of the recorded videos, we observed students having difficulties navigating the tools' menus and options, forcing them to attempt the same task several times before finding the proper way to perform their goal. As explicitly evident in most of the videos, this generated a considerable feeling of frustration among them.

To overcome the modeling obstacles detected in our study (and indirectly decrease the modeling effort), we suggest several recommendations mainly oriented to improve the usability of modeling tools:

- First, and most important, modeling tools should fix their current known issues to increase the level of trustiness in them. For instance, in the Neon version of Papyrus, it is not possible to introduce the multiplicity of an association class created using the *AssociationClass* element in the palette.
- Modeling tools should include a guidance for beginners to perform specific tasks. This guidance should be available under user request, for instance, as short embedded videos showing how to carry out specific tasks using the tool (such as creating association classes, etc.). Modeling wizards could also be created for this purpose.
- Modeling tools should provide useful tips not only when the tool is opened but when the tool detects the modeler really needs such tip. For instance, tools could detect when the modeler is in a loop trying to achieve a specific task and suggest useful and personalized tips to help her. AI techniques could be employed to detect such tool learning challenges and react accordingly.

Although most of the obstacles observed are directly related with the modeling tools, like the ones above, other obstacles are more linked to a general lack of modeling knowledge. To improve them, we would like to see some tweaks on how we teach modeling:

- Modeling instructors should explain that some elements in a class diagram should be created following a specific order, even if it is just for pragmatic reasons.
- Instructors should also emphasize the use of the best practices in conceptual modeling, as those detailed by Kuzniarz et al.[27].
- And they should recommend the use of proper names to facilitate the understandability of the class diagrams, for instance making use of the naming guides provided by Aguilera et at [28]. As an

Table 12
Estimated time to build the UML class diagram using MagicDraw and Papyrus.

Sub-goal	Occurrences in the class diagram	Estimated time in MagicDraw (sec)	Estimated time in Papyrus (sec)
Create a new project	1	6.80	12.95
Create an empty class Diagram	1	5.83	16.30
Create a UML class (with name)	5	31.25	28.00
Add an attribute (with name and type) to an existing class	8	65.50	75.20
Create a binary association (with name and multiplicity) between two existing classes	2	23.2	20.40
Create an association class (with name and multiplicity) between two existing classes	1	15.10	15.60
	Total estimated time (sec)	**147.78 (sec)**	**168.45 (sec)**
	Total estimated time (min)	**2.46 (min)**	**2.81 (min)**

example, one naming guideline from the previous work states that "the name of an entity type should be a noun phrase whose head is a countable noun in singular form". For instance, names such as *Person, Chair, Invoice* or *Category* (classes), *Job* or *Enrollment* (association classes), *Date* or *AmountOfMoney* (data types), and *Sex* or *Color* (enumerations) follow the guideline.

8. Conclusions and further work

In this paper we have analyzed how UML class diagrams are built by analyzing video recordings of 45 students in the process of creating class diagrams using the MagicDraw and Papyrus tools in a controlled experiment. We evaluated the over 12 h of videos to analyze the usability of such tools from three perspectives: the modeling process, the modeling effort and the modeling obstacles they encounter during the process. To deepen the study of the modeling effort perspective, we carried out a complementary analytical study comparing the efficiency (in terms of number of clicks and time) of both tools during the performance of several tasks required to build a UML class diagram by an expert modeler.

Our experiments report that there are no notable differences regarding he usability of MagicDraw and Papyrus. But only because both showed important shortcomings that could be improved to provide a significantly better modeling experience. Several suggestions related to these three perspectives have been provided.

As a further work, we plan to extend this study considering other UML modeling tools (not only graphical tools but also textual tools like TextUML or Umple, including a grouped comparison of graphical vs textual tools) and UML diagrams to see how the results of this study can be generalized. We would also like to discuss with UML tool builders to try to integrate some of our suggestions in their future releases and to replicate our study in other modeling fields, like BIM (Building Information Modeling) and see if the modeling tools in those domains suffer from the same problems (and therefore could also benefit from an adapted version of our suggestions).

Conflict of interest

The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

CRediT authorship contribution statement

Elena Planas: Investigation. **Jordi Cabot:** Investigation.

Acknowledgments

This work has been supported by H2020-ICT-2015/H2020-ICT-2015 TeSLA project "An Adaptive Trust-based e-assessment System for Learning", Number 688520. This work has also received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737494. A special thank to all the students who made possible our experiment, Javier Luis Cánovas Izquierdo for its help in the statistical part and all the reviewers for their useful comments on the previous drafts of this paper.

References

[1] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, 1st, Morgan & Claypool Publishers, 2012.
[2] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri, E. Tosetti, Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA, Web Engineering, 7th International Conference, ICWE 2007, Como, Italy, July 16-20, 2007, Proceedings, (2007), pp. 539–544.
[3] R.B. France, A. Evans, K. Lano, B. Rumpe, The UML as a formal modeling notation, Comput. Stand. Interf. 19 (7) (1998) 325–334.
[4] L.T.W. Agner, T.C. Lethbridge, A survey of tool use in modeling education, 20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, Austin, TX, USA, September 17-22, 2017, (2017), pp. 303–311.
[5] M. Auer, L. Meyer, S. Biffl, Explorative UML modeling - comparing the usability of UML tools, ICEIS 2007 - Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume EIS, Funchal, Madeira, Portugal, June 12-16, 2007, (2007), pp. 466–473.
[6] A.E. Bobkowska, K. Reszke, Usability of UML modeling tools, Software Engineering: Evolution and Emerging Technologies, (2005), pp. 75–86.
[7] M.R.V. Chaudron, W. Heijstek, A. Nugroho, How effective is UML modeling ? - an empirical perspective on costs and benefits, Softw. Syst. Model. 11 (4) (2012) 571–580.
[8] I. Davies, P.F. Green, M. Rosemann, M. Indulska, S. Gallo, How do practitioners use conceptual modeling in practice? Data Knowl. Eng. 58 (3) (2006) 358–380.
[9] B. Dobing, J. Parsons, How UML is used, Commun. ACM 49 (5) (2006) 109–113.
[10] H. Eichelberger, Y. Eldogan, K. Schmid, A comprehensive survey of UML compliance in current modelling tools, Software Engineering 2009: Fachtagung des GI-Fachbereichs Softwaretechnik 02.-06.03. 2009 in Kaiserslautern, (2009), pp. 39–50.
[11] Heena, R. Garg, A comparative study of UML tools, International Conference on Advances in Computing and Artificial Intelligence, ACAI '11, Rajpura/Punjab, India - July 21, - 22, 2011, (2011), pp. 1–4.
[12] L. Khaled, A comparison between UML tools, 2009 Second International Conference on Environmental and Computer Science, ICECS 2009, Dubai, UAE, 28–30 December 2009, (2009), pp. 111–114.
[13] M. Petre, UML in practice, 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18–26, 2013, (2013), pp. 722–731.
[14] J. Recker, "Modeling with tools is easier, believe me" - The effects of tool functionality on modeling grammar usage beliefs, Inf. Syst. 37 (3) (2012) 213–226.
[15] G. Reggio, M. Leotta, F. Ricca, D. Clerissi, What are the used UML diagrams? A preliminary survey, Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013), Miami, USA, October 1, 2013. (2013), pp. 3–12.
[16] S.A. Safdar, M.Z. Iqbal, M.U. Khan, Empirical evaluation of UML modeling tools-a controlled experiment, Modelling Foundations and Applications - 11th European Conference, ECMFA 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 20–24, 2015. Proceedings, (2015), pp. 33–44.
[17] K. Siau, P. Loo, Identifying difficulties in learning UML, IS Manag. 23 (3) (2006) 43–51.
[18] B. Dobing, J. Parsons, How UML is used, Commun. ACM 49 (5) (2006) 109–113.
[19] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.
[20] F. Ricca, M.D. Penta, M. Torchiano, P. Tonella, M. Ceccato, How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: A series of four experiments, IEEE Trans. Software Eng. 36 (1) (2010) 96–118.
[21] M. Torchiano, F. Ricca, P. Tonella, Empirical comparison of graphical and annotation-based re-documentation approaches, IET Softw. 4 (1) (2010) 15–31.
[22] A. Yamashita, F. Petrillo, F. Khomh, Y.-G. Guhneuc, Developer Interaction Traces backed by IDE Screen Recordings from Think aloud Sessions, MSR 2018 - 15th International Conference on Mining Software Repositories, co-located with ICSE 2018 in Gothenburg, Sweden. (2018), pp. 50–53.
[23] G.C. Murphy, M. Kersten, L. Findlater, How Are Java Software Developers Using the Eclipse IDE? IEEE Softw. 23 (4) (2006) 76–83.
[24] K. Thulasiraman, M.N. Swamy, Graphs: Theory and Algorithms, John Wiley & Sons, 2011.
[25] B.E. John, D.E. Kieras, Using GOMS for user interface design and evaluation: which technique? ACM Trans. Comput.-Hum. Interact. 3 (4) (1996) 287–319.
[26] D. E. Kieras, Using the Keystroke-Level Model to Estimate Execution Times, University of Michigan 555 (2001).
[27] L. Kuzniarz, M. Staron, Best practices for teaching UML based software development, Satellite Events at the MoDELS 2005 Conference, MoDELS 2005 International Workshops, Doctoral Symposium, Educators Symposium, Montego Bay, Jamaica, October 2–7, 2005, Revised Selected Papers, (2005), pp. 320–332.
[28] D. Aguilera, C. Gómez, A. Olivé, A complete set of guidelines for naming UML conceptual schema elements, Data Knowl. Eng. 88 (2013) 60–74.