Comparing UML-based and DSL-based Modeling from Subjective and Objective Perspectives *

África Domingo[1], Jorge Echeverría[1], Óscar Pastor[2], and Carlos Cetina[1]

[1] Universidad San Jorge. SVIT Research Group, Zaragoza, Spain
{adomingo,jecheverria,ccetina}@usj.es
[2] Universidad Politecnica de Valencia. PROS Research Center, Valencia, Spain
opastor@dsic.upv.es

Abstract. In the last two decades, researchers have conducted several empirical evaluations, involving thousands of subjects, to understand the use of models in software development. The results of these evaluations show that most of the subjects make informal use of the models, which is known as 'modeling as sketch'. In this paper, we present an experiment that compares UML-based and DSL-based modeling when subjects model a part of a commercial video game. In the comparison, we have used objective and subjective measures, in contrast to other works that focus either on objective measures to evaluate modeling performance or on subjective measures to analyze modeling styles. Our results reveal that subjects underestimate the potential of their own models. Our finding is relevant for the design of future evaluations and for the teaching and adoption of modeling. If users correctly assess their models, they might leverage their potential as programs.

Keywords: Empirical comparison · Modeling languages · UML · DSL

1 Introduction

The software engineering process goes from the initial idea for building a particular software (problem space) to the code that implements the software (solution space) through different transformations. The idea is first conceptualized through models, and then the models are transformed into fully functional code. To understand the use of modeling languages in software development, in the last two decades, researchers have conducted surveys [17, 13, 1, 16, 20, 27, 2, 8], case studies [4, 26, 8], experiments [31, 23, 25, 11, 5, 8], and interviews [8] involving more than 5,355 subjects. The researchers classified the styles of modeling into three categories: *sketch* (informal models to aid in communication, see Figure 1.a), *blueprint* (models as the basis for programmers to create code, see Figure 1.b), and *programs* (models that include all of the details needed to generate code, see

* Partially supported by MINECO under the Project ALPS (RTI2018-096411-B-I00).

Figure 1.c). All of the works that evaluated the style of modeling [17, 8, 16, 27, 2] concur that most of the subjects make an informal use of the models, which is known as 'modeling as sketch'. Despite the benefits of using models as programs [14], models are still mainly used as sketch.

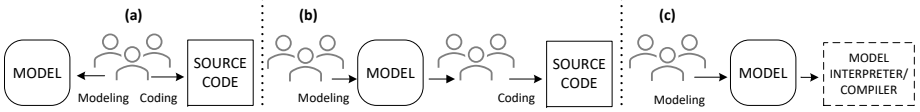

Fig. 1. (a) Models as sketch, (b) Models as blueprint, (c) Models as programs

The Unified Modeling Language (UML) is a general-purpose modeling language that has become the 'de facto' standard for modeling software systems [17, 7, 8]. Despite the available evidence about the efficiency of UML modeling, UML shortcomings have been identified by several authors [17, 4, 13, 26, 7, 8, 1]. In addition, to reduce the complexity of software development, the use of domain-specific languages (DSLs), languages closer to the problem domain, has proven to outperform the use of general-purpose programming languages [19]. We can also find empirical works that indicate that the use of a DSL can reduce the effort of developers when working in modeling performance testing tasks [5].

In empirical research, comparing UML against DSL is an open problem towards understanding software modeling [22, 7]. Previous works [19, 5] suggest that the usage of DSLs outperforms the usage of other software artifacts. Our own previous work in the field [14] points in the same direction. Thus far, works that have compared UML against other modeling languages [5, 11, 25, 23, 31] have focused solely on objective measures. In this work, we combine objective measures (correctness and efficiency) with the subjective assessment of subjects (model style classification and satisfaction) through a crossover experiment that compares UML vs DSL when subjects model an element of a commercial video game. Game software engineering has been identified as a knowledge area that needs more fundamental research [3]. Moreover, despite video-game development being one of the fastest growing industries, there are few works in the field, which makes it an original domain for exploring modeling adoption.

Building upon previous work we hypothesize that the usage of a DSL will outperform the usage of UML models in the video-games domain. The findings of this empirical study not only confirm this hypothesis, but in addition, reveal a problem that was not found in evaluations performed by other authors: no matter how correct the models are, and their ability to actually generate software, the subjects think of them more as sketches than as programs.Our results are useful for teaching modeling and model adoption because they put the focus on the problem of subjects underestimating the potential of their own models.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 describes our experiment, and Section 4 shows the results. Section 5 describes the threats to validity. Finally, Section 6 concludes the paper.

2 Related Work

Modeling languages have been identified as a key challenge to increase the adoption of models as programs [22], and the use of DSLs has been identified as more effective and efficient than general-purpose programming languages [19]. However, it is difficult to find empirical works that compare UML with an alternative modeling language. Budgen [7] explained in 2011 that this was because of the underlying assumption that UML did not require testing because it was a de facto standard. The oldest comparison work we have found dates from 2001. Zendler et al. [31] compared three object-oriented approaches (UML, Open modeling language [OML], and taxonomic object system [TOS]) in two different application systems with respect to coarse-grained modeling concepts. They conclude that the coarse-grained concepts of the object-oriented approaches OML and TOS were superior to those of UML when modeling a database-oriented application.

In 2004, Otero and Dolado [23] presented the results of a controlled experiment comparing the comprehension of UML and OML diagrams in the design of a real-time embedded system. They found that the specification of dynamic data is faster and easier to comprehend in OML than in UML. A year later, Iris Reinhartz-Berger and Dov Dori [25] compared UML with Object-Process Methodology (OPM) with respect to the level of comprehension and the quality of Web application models. The results of their experiment suggest that OPM is better than UML when modeling web applications, especially in the dynamics aspects and in the quality of the models.

In 2010, De Lucia et al.[11] presented the results of three sets of controlled experiments comparing UML class diagrams and Entity-Relationship (ER) diagrams with respect to the comprehension and the interpretation of data models during maintenance activities. The results demonstrated that the two notations gave the same support, except during verification activities when UML class diagrams provided better support than ER diagrams. In 2016, Bernardino et al. [5] presented the results of an experiment about the benefits and drawbacks when using UML or a DSL for modeling performance testing in an IT company. Their results indicate that the effort using a DSL was lower than using UML.

Table 1 shows the works on adopting models in the industrial context. Modeling as sketch (also known as informal modeling) is the second problem that has been identified most frequently in the literature [17, 8, 16, 27, 2] (see P1 in Table 1). Other studies have identified other problems such as the lack of understanding [17, 13, 1] (see P2) or training [4, 13, 1, 2] (see P3), or the lack of integration between modeling tools [4, 26, 8, 1, 20, 2] (see P4 in Table 1).

Through a survey, Grossman et al. [17] investigated if individuals who use UML perceive it to be beneficial and which characteristics affect the use of UML. They used the Task Technology Fit index to evaluate the respondents' perceptions. They affirmed that most of the respondents of the survey were using what Fowler [15] calls 'UML by sketch', an informal approach to modeling and the values of TTF index indicated a slightly positive perception of UML. Chaudron et al. [8] synthesized a selection of empirical evidence (one experiment, one Case Study, two Surveys, and more than 20 Interviews) about the efficiency of UML

Table 1. Empirical studies on modeling adoption

Work Year	Empirical strategy	Sample size	Context	Modeling language	Main problems identified
Grossman et al. [17] 2004	Survey	131 UML users	Industry	UML	UML as sketch is the most used style of UML modeling (P1) Lack of adequate UML understanding (P2)
Anda et al. [4] 2006	Case Study	16 System developers and project manager	Industry	UML	Inadequate level of UML training (P3) Inadequate modeling tools (P4)
Dobing and Parson [13] 2006	Survey	171 annalist using UML	Industry	UML	Lack of adequate UML understanding (P2) Inadequate offer of UML training (P3)
Staron [26] 2006	Case Study	8 Professionals	Industry	UML	Lack of well-integrated tools (P4)
Chaudron et al. [8] 2012	1 Experiment, 1 Case Study, 2 Surveys, 20+ Interviews	200+ Professionals and students	Industry Academia	UML	Modeling as sketch and for communication are the most used styles of modeling (P1) Lack of well-integrated tools (P4)
Agner et al. [1] 2013	Survey	209 Embedded-industry developers	Industry	UML	Lack of adequate UML understanding (P2) Lack of specialized professionals (P3) Inappropriate tool support (P4)
Gorschek et al. [16] 2014	Survey	3785 Developers	Industry	All	Models are used primarily in communication and collaboration (P1) Inadequate modeling tools (P4)
Marko et al. [20] 2014	Survey	112 Embedded-industry developers	Industry	All	Lack of well-integrated tools (P4)
Störrle [27] 2017	Survey	96 Professionals	Industry	All	Modeling as sketch is the most used style of modeling (P1) Cultural differences in modeling usage (P6)
Akdur et al. [2] 2018	Survey	627 Embedded-industry developers	Industry	All	Modeling as sketch is the most used style of modeling (P1) Lack of modeling expertise(P3) Inappropriate tool support (P4)

modeling in software development. They concluded that especially for larger and distributed projects, UML is used to understand a problem at an abstract level and to share information with other team members. In these cases, UML was used without rigor and specialized tools were not used for the modeling.

Gorschek et al. [16] presented the results of a survey summarizing the answers of 3785 developers to a simple question: Which design models are used before coding? The answer was that design models were not used very extensively in industry and that when they were used, their use was informal and without tool support, and the notation was often not UML. Again, they found that models were used primarily as a communication and collaboration mechanism, where there is a need to solve problems or to obtain a joint understanding of the overall design by a team. Störrle [27] presented the results of an online survey, with 96 industry participants from all over the world. The questions in this case were 'How and what are the models used for?' He found that models were widely used in industry and that UML was indeed the leading language. This directly contradicts the results of Gorschek et al.[16]. He reported three distinct usage modes of models, the most frequent of which was informal usage for communication and understanding, and program-style [24] usage was rare. A year later, in 2018, Akdur et al.[2] conducted another online survey with opinions of 627 practicing

embedded software engineers from 27 different countries. The survey addressed the state of software modeling and MDE practices in the worldwide embedded software industry. Their results match those of H. Störrle [27]:the majority of participants were using UML, and its use was informal.

Our work includes a quantitative perspective to discuss the differences in performance of the modeling languages, but we also take into account the modeling style in the analysis to reach a more complete understanding of modeling usage. Furthermore, the classification of models in previous works [17, 8, 16, 27, 2] is taxonomic; the use that subjects made of models is classified into one style or another in accordance with their responses to certain questions in surveys or interviews. In our experiment, the subjects evaluate their models according to the usefulness of the model for each one of the modeling styles (sketch, blueprint or programs). This offers more complete information on the perception that the subjects have of the usefulness of their models.

3 Experiment design

3.1 Objectives

According to the guidelines for reporting software engineering experiments [30], we have organized our research objectives using the Goal Question Metric template for goal definition. Our goal is to **analyze** modeling languages and their perceived usefulness, **for the purpose of** comparison, **with respect to** correctness of the models constructed, efficiency, and user satisfaction, **from the point of view of** novice and professional developers, **in the context of** modeling for a video-game company.

3.2 Variables

In this study, the factor under investigation is the *Modeling Language*. There are two alternatives, UML and DSL, which are the modeling languages used by subjects to model an enemy of a commercial video game.

Since the goal of this experiment is to evaluate the effects of the use of different modeling languages, we selected *Correctness* and *Efficiency* as the objective response variables, which are related to modeling performance. We measured *Correctness* using a correction template, which was applied to the models developed by the subjects after the experiment. To calculate *Efficiency*, we measured the time employed by each subject to finish the task, using the start and end time of each task. *Efficiency* is the ratio of *Correctness* to time spent (in minutes) to perform a task.

We also compared UML and the DSL with respect to *Satisfaction* using a 5-point Likert-scale questionnaire based on the Technology Acceptance Model (TAM) [21].

We decompose *Satisfaction* into three subjective response variables as follows: *Perceived Ease of Use* (PEOU), the degree to which a person believes that

learning and using a particular language would require less effort. *Perceived Usefulness* (PU), the degree to which a person believes that using a particular language will increase performance, and *Intention to Use* (ITU), the degree to which a person intends to use a modeling language. Each of these variables corresponds to specific items in the TAM questionnaire. We average the scores obtained for these items to obtain the value for each variable.

To analyze the subjective perception of the models, the subjects evaluated the usefulness of their models using a 5-point Likert-scale for each style of modeling: to understand and communicate (sketch), for developers to create code (blueprint) and to automatically generate code (programs).

3.3 Design

We chose a factorial crossover design with two periods using two different tasks, T1 and T2, one for each period. All of the subjects used the two modeling languages, each one of which was used in a different task. The subjects had been randomly divided into two groups (G1 and G2). In the first period of the experiment, all of the subjects solved T1 with G1 using UML and G2 using DSL. Afterwards, all of the subjects solved T2, G1 using DSL and G2 using UML.

This repeated measures design increases the sensitivity of the experiment [28]: the observation of the same subject using the two alternatives controls between-subject differences, improving experiment robustness regarding variation among subjects. By using two different sequences for each group (G1 used UML first and DSL afterwards, and G2 used DSL first and UML afterwards) and different tasks, the design counterbalances some of the effects caused by using the alternatives of the factor in a specific order (i.e., learning effect, fatigue). To verify the experiment design, we conducted a pilot study with two subjects. The subjects in the pilot study did not participate in the experiment.

3.4 Research questions and hypotheses

The research questions and null hypotheses are formulated as follows:

RQ1 Does the modeling language used for modeling software impact the *Correctness* of the models? The corresponding null hypothesis is $H_{0,C}$: The modeling language does not have an effect on *Correctness*.

RQ2 Does the modeling language used for modeling software impact the *Efficiency* of developers to model? The null hypothesis for *Efficiency* is $H_{0,E}$: The modeling language does not have an effect on *Efficiency*.

RQ3 Is the user satisfaction different when developers use different modeling languages? To answer this question, we formulated three hypotheses based on the variables *Perceived Ease of Use*, *Perceived Usefulness*, and *Intention to Use*, with their corresponding null hypotheses. These are: $H_{0,PEOU}$, The modeling language does not have an effect on *Perceived Ease of Use*; $H_{0,PU}$, The modeling langu age does not have an effect on *Perceived Usefulness*; $H_{0,ITU}$, The modeling language does not have an effect on *Intention to Use*.

The hypotheses are formulated as two-tailed hypotheses.

3.5 Participants

The subjects were selected using convenience sampling [30]. We invited 26 third-year undergraduate students (novices) from a technology program who had passed previous courses where UML modeling was analyzed and used. Of these subjects, 25 decided to participate and 22 completed the tasks and forms. We also invite 17 professionals who are linked to modeling or video-game development to participate in the experiment. Nine of them decided to participate and completed the experiment. A total of 31 subjects with different knowledge about modeling performed the experiment.

The subjects filled out a demographic questionnaire that was used for characterizing the sample. Table 2 shows the mean and standard deviation of age, hours per day developing software (Developing time), and hours per day working with models (Modeling time). We used a 5-point Likert-scale to measure the subjects' knowledge of programming languages (Programming knowledge), modeling languages (Modeling Knowledge) and domain-specific languages (DSL knowledge). The mean and standard deviation of their answers are also shown in Table 2. The subjects recognized having a medium-high knowledge about software modeling or specific domain languages. All of them spent more time coding than modeling and evaluated their programming language knowledge or their ability with models higher than their knowledge about domain-specific languages.

Table 2. Results of the demographic questionnaire

	Age $\mu \pm \sigma$	**Developing time**$\pm\sigma$	**Modeling time**$\pm\sigma$	**Programming known**$\pm\sigma$	**Modeling known**$\pm\sigma$	**DSL known**$\pm\sigma$
All subjects	23.1±4.5	2.5±2.4	0.9±1.1	3.8±0.9	3.0±1.1	2.6±1.1
Novices	21.1±1.8	1.6±1.7	0.7± 0.8	3.5±0.9	2.5±0.8	2.2±0.9
Professionals	27,9±5.4	4.8±2.2	1.3±1.5	4.4±0.5	4.0±0.8	3.7±0.9

The experiment was conducted by two instructors and one expert in the video-game software domain. The expert provided information about the domain and about the Kromaia DSL. This expert was not the same person who was responsible for designing the tasks. During the experiment, one of the instructors gave the instructions and managed the focus groups. The other instructor clarified doubts about the experiment and took notes during the focus group.

3.6 Experimental objects and procedure

The tasks of our experiment were extracted from a real-world software development, Kromaia, which is a commercial video game released on PlayStation 4 and Steam. In Kromaia, the models are interpreted at run-time to create the C++ games' objects [6]. For modeling, the subjects used Shooter Definition Modeling

Language, which is the DSL used in Kromaia[3] and UML. A video-game software engineer, involved in the development of Kromaia, designed the two tasks of similar difficulty and prepared the correction template for the DSL task. An expert on modeling prepared the correction template for the UML task. The experimental objects used in this experiment (which includes the training material, the tasks and the forms used for the questionnaires) as well as the results and the statistical analysis are available at http://svit.usj.es/UMLvsDSL-experiment.

The experiment was conducted on-line due to the COVID19 pandemic restrictions. During the experiment, all of the participants joined the same videoconference via Microsoft Teams, and the chat-session was used to clarify doubts or share information. Two forms were prepared on Microsoft Forms for data collection, one for each experimental sequence. The experiment, scheduled for 2 hours, was conducted on two different days with different groups. On the first day, it was performed by novices and on the second day it was performed by professionals. The experimental procedure is described as follows:

1. An instructor explained the parts in the session, and he advised to the subjects that it was not a test of their abilities.
2. The subjects attended a tutorial about the video-game enemies to be modeled and about the DSL used in the experiment. The information used in the tutorial and a UML usage guide were available to the subjects during the experiment. The time spent on this tutorial was less than 10 minutes.
3. The subjects received clear instructions on where to find the links to access the forms for the experiment. They were also told about the structure of these forms and where they could find information about UML and the DSL if they needed to. The subjects were randomly divided into two groups (G1 and G2); the subjects from G1 received the links to access one form and the subjects from G2 received a link to another. form.
4. The subjects accessed to the on-line form and read and confirmed having read the information about the experiment, the data treatment of their personal information, and the voluntary nature of their participation before accessing the questionnaires and tasks of the experiment.
5. The subjects completed a demographic questionnaire.
6. The subjects performed the first task. The subjects from G1 had to use DSL to model a video-game enemy, and the subjects from G2 had to use UML to model the same enemy. After submitting their solution, the subjects classified the model they had built according to its usefulness, and they completed a satisfaction questionnaire about the modeling language used.
7. The subjects performed the second task. The subjects from G1 modeled another video-game enemy using UML, and the subjects from G2 modeled the same enemy using the DSL. Then, the subjects classified the model they had built and completed the satisfaction questionnaire.
8. A focus group interview about the tasks (with average duration of 15 minutes) was conducted by one instructor while the other instructor took notes.
9. Finally, the tasks were corrected and a researcher analyzed the results.

[3] Learn more of Kromaia DSL at: https://youtu.be/Vp3Zt4qXkoY

3.7 Analysis procedure

We have chosen the Linear Mixed Model (LMM) [29] for the statistical data analysis. LMM handles correlated data resulting from repeated measurements, and it allows us to study the effects of factors that intervene in a crossover design (period, sequence, or subject) and effects of other blocking variables (e.g., in our experiment, professional experience) [28].

In this study, we apply the Type III test of fixed effects with unstructured repeated covariance. The *Modeling Language*(ML) was defined as a fixed-repeated factor to identify the differences between using UML or DSL, and the subjects were defined as random factor $(1|Subject)$ to reflect the repeated measures design. The response variables (RV) for this test were *Correctness* and *Efficiency*, and the three other variables correspond to *Satisfaction*: *Perceived Ease of Use* (PEOU), *Perceived Usefulness* (PU) and *Intention to use* (ITU).

The assumption for applying LMM is normality of the residuals of the response variables. To verify this normality, we used Kolmogorov-Smirnov tests as well as visual inspections of the histogram and normal Q-Q plots.

The starting statistical model (Model 0) to be tested reflects the principal factors used in this experiment and is described as:

$$RV \sim ML + (1|\, subject)\ (Model\ 0) \tag{1}$$

In order to take into account the potential effects of factors that intervene in a crossover design in determining the main effect of *Modeling Language*, we considered *Period* and *Sequence* to be fixed effects. We also considered fixed factors that are related to the subject's experience in the statistical model in order to explore the potential effects of *Experience* or the effects of the sequence *Modeling Language* and *Experience* ($ML * Experience$) to determine the variability in the response variables. We tested different statistical models like the ones used in the following formulas in order to find out which factors, in addition to *Modeling Language*, could best explain the changes in the response variables:

$$\begin{array}{ll} RV \sim ML + Experience + (1|\, Subject) & (Model\ 1) \\ RV \sim ML + Experience + ML * Experience + (1|\, Subject) & (Model\ 2) \\ RV \sim ML + Experience + Period + (1|\, Subject) & (Model\ 3) \end{array} \tag{2}$$

The statistical model fit of the tested models was evaluated based on goodness of fit measures such as Akaike's information criterion (AIC) and Schwarz's Bayesian Information Criterion (BIC). The model with the smallest AIC or BIC is considered to be the best fitting model [18]. To describes the changes in each response variable we selected the statistical model that satisfied the normality of residuals and also obtained the smallest AIC or BIC value.

To quantify the differences in the response variables due to significant fixed factors, we calculated the the Cohen d value [9] between the alternatives of these factors. Values of Cohen d between 0.2 and 0.3 indicate a small effect, values around of 0.5 indicate a medium effect, and values greater than 0.8 indicate a large effect. We selected histograms and box plots to graphically describe the data and the results.

4 Results

4.1 Changes in the response variables

There were differences in the means of all of the response variables depending on which *Modeling Language* was used to model a video-game enemy. Table 3 shows the values for the mean and standard deviation of the dependent variables *Correction*, *Efficiency*, *Perceived Ease of Use* (PEOU), *Perceived Usefulness* (PU), and *Intention to use* (ITU) for each one of the *Modeling Languages* compared: UML and DSL.

Table 3. Values for the mean and standard deviation of the response variables

	Correctness	*Effectiveness*	*Satisfaction* $\mu \pm \sigma$		
	μ % $\pm\sigma$	μ %/*min* $\pm\sigma$	*PEOU*	*PU*	*ITU*
DSL	90.32 % ± 12.36	6.16 %/*min* ± 2.91	4.27 ± 0.61	4.11 ± 0.63	3.9 ± 0.99
UML	71.87 % ± 25.16	4.10 %/*min* ± 3.51	2.85 ± 0.95	2.83 ± 1.02	2.45 ± 1.23

According to the Cohen d values of the response variables, we can affirm that the effect of the *Modeling Language* on *Correctness* is large, with a Cohen d value of 0.930 and that the effect on *Efficiency* is medium-large, with a Cohen d value of 0.639. The effect size for *Satisfaction* is very large, with Cohen d values of 1.786, 1.515, and 1.327 for *Perceived Ease of Use*, *Perceived Usefulness*, and *Intention to use*, respectively. These values are related to the percentage of non-overlap between the distributions of the response variables for each modeling language. Higher values correspond with greater percentages of non-overlap and larger differences. The histograms of Figure 2 illustrate the differences in the response variables. In the histograms, the non-overlapping parts have a single pattern (either dots or shaded), while the overlapping parts have both patterns (dots and shaded).

For all of the variables, the factor *Modeling Language* obtained p-values of less than 0.05, regardless of the statistical model used for its calculation. Therefore, all of the null hypotheses are rejected. Thus, the answers to the research questions **RQ1**, **RQ2**, and **RQ3** are affirmative. The Modeling Language used for developing software has a significant impact on the correctness of the model, efficiency, and the satisfaction of software developers.

For *Correctness*, the chosen statistical model was Model 1 (See formula (2)), which was the best fitting model that satisfied normality in the residuals of the response variable. The fixed factors, *Modeling Language* (F=15.156, p=0.001) and *Experience* (F=4.393, p=0.045), were considered to be statistically significant to explain the changes in correctness. The Cohen d value of -0.654 calculated with the standardized difference between the means of *Correctness* for novices and professionals indicates a medium-large effect in favour of the professionals. The models made by professionals are more correct than the ones made by novices.

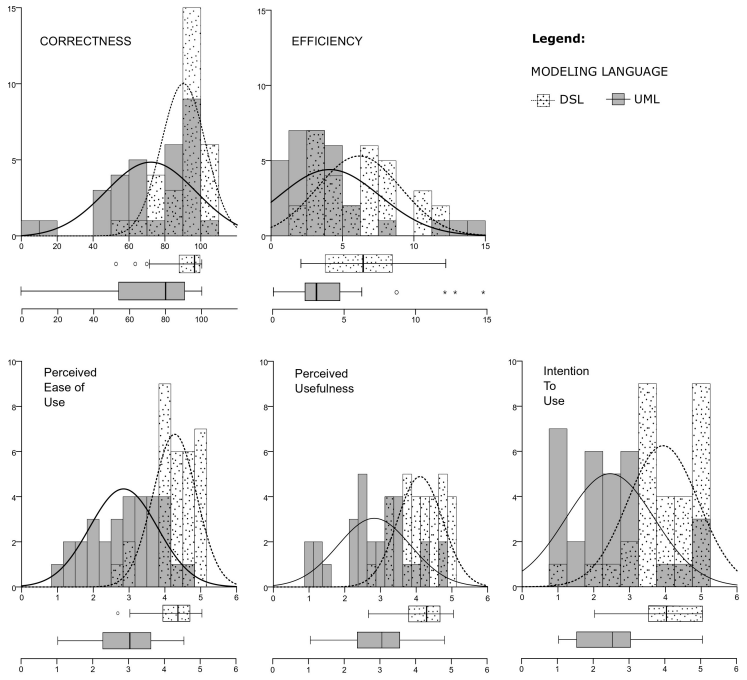

Fig. 2. Histograms with normal distributions and box plots for the response variables

For *Efficiency* the chosen statistical model was Model 3 (See formula (2)). The fixed factors *Modeling Language* (F=12.337, p=0.001), *Experience* (F=5.275, p=0.029), and *Period* (F=16.451, p=0.000) were considered to be statistically significant to explain the changes in correctness. The Cohen d value of -0.592 for *Efficiency* between novices and professionals indicates a medium-large effect in favour of the professionals. The professionals are more efficient than novices when modeling. The Cohen d value of -0.788 for *Efficiency* between the first period and the second period, indicates a large effect in favour of the second period. The subjects were more efficient when modeling the second task.

For *Satisfaction* the *Modeling Language* factor was the one fixed factor that was found to be significant for all the statistical models tested. For *Perceived Ease of Use* and *Perceived Usefulness*, the chosen model was the starting statistical model, Model 0 (See formula (1)) and the *Modeling Language* factor obtained p-values of 0.000 for *Perceived Ease of Use* (F=963.137, p=0.000) and *Perceived Usefulness* (F=892.660, p=0.000). For *Intention to use*, the chosen model was Model 2 (See formula (2)); however, only the *Modeling Language* factor (F=13.846, p=0.001) was statistically significant in explaining the changes in this response variable. The changes in *Intention to use* due to *Experience* (F=0.604, p=0.443) or the sequence *Modeling Language* and *Experience* ($ML{*}Experience$) (F=0.115, p=0.737) were not considered to be statistically significant.

4.2 Model assessment by subjects

All of the subjects evaluated their DSL models better than their UML models. The novices considered their models to be more useful as sketch than as blueprint or as programs. The professionals evaluated the usefulness of their DSL model better as programs than as sketch or blueprint, and they evaluated their UML model to be more useful as sketch than as blueprint or as programs. Table 4 shows the values for the mean and standard deviation of the subjects' evaluation of their own models for each modeling style.

Table 4. Values for the mean and standard deviation for models usefulness

		As sketch $\mu \pm \sigma$	*As blueprint* $\mu \pm \sigma$	*As a program* $\mu \pm \sigma$
DSL	Novices	4.1 ± 1.0	3.9 ± 1.1	3.5 ± 1.4
	Professionals	4.2 ± 0.8	3.7 ± 0.8	4.3 ± 0.5
UML	Novices	3.0 ± 1.0	3.3 ± 0.8	2.6 ± 1.1
	Professionals	3.8 ± 1.0	3.0 ± 1.1	2.9 ± 1.2

During the focus group, the novices declared that they found the DSL to be useful for communication, but they believed that UML would be necessary to generate a video-game enemy with all of its characteristics. The professionals (especially those linked to video games) said that UML could be useful to define a video-game enemy in general, at the beginning of the development, but that they would choose a DSL to define a specific video-game enemy.

4.3 Interpretation of the results

During the training of the experiment, the subjects were introduced to Kromaia, and the use of models made in Kromaia was explained to them. In Kromaia, models are used as programs (see Figure 1.c). It is very striking that, despite the fact that a successful case of models as programs was used and that the subjects were explicitly told that Kromaia developers use models as programs, most of the subjects continue to view their models as sketch rather than as programs.

Furthermore, the results also show that models are not far from scoring a value of 100% correct. On average, only 28% would have to be corrected in UML models and 10% in DSL models to be used as programs. Actually, more than five subjects produced models with a value of 100% correct. These models can be run on top of the Kromaia model interpreter. Nevertheless, they did not give a higher score to programs than to sketch. This is a problem that was not covered in evaluations by others authors: no matter how correct the models are, the subjects value them more as sketches than as programs. We suggest that this is a major problem: it is not that the models are not ready to be used as programs because we must improve tools and teaching, it is that subjects think of models more as sketches than as programs.

5 Threats To Validity

To describe the threats to validity of our work, we use the classification of [30]:

Conclusion validity. The *low statistical power* threat was minimized because the confidence interval is 95%. The *fishing and the error rate* threat was minimized using tasks and fixes designed by a video-game software engineer. The *Reliability of measures* threat was mitigated because the objective measurements were obtained from the digital artifacts generated by the subjects when they performed the tasks. The *reliability of treatment implementation* threat was alleviated because the procedure was identical in the two sessions. Also, the tasks were designed with similar difficulty.

Internal validity. The *compensatory rivalry* threat affected the experiment; the subjects may have been motivated to perform the task with a higher level of quality by using the modeling language that was the most familiar to them. The *interactions with selection* threat affected the experiment because of the voluntary nature of participation. To avoid student demotivation, we selected students of a course whose contents fit the design of the experiment. In addition, the subjects had different levels of modeling language knowledge and different levels of knowledge of the video-game domain. To mitigate this threat, the treatment was applied randomly. However, we found two outliers during the analysis of correctness and efficiency. The extreme values that were found correspond to subjects with scores of less than 25% in correctness in the UML task. Following the recommendations of Dean et al. [12], we repeated the statistical analysis by excluding the data of these two subjects, and we found that the language factor remained statistically significant to explain the changes in all of the response variables for all of the models tested. Hence, our conclusions were not sensitive to the responses of subjects with low scores in correctness. Even though the tasks were designed with similar complexity, the effect of the task (period) was significant for efficiency. The effect of the language being the same for both tasks suggests a learning effect; the subjects spent less time performing the second task. We minimized this *maturation* threat by using a crossover design.
Construct validity. All of the measurements were affected by *Mono-method bias*. To mitigate this threat for the correctness and efficiency measurements, we mechanized these measurements as much as possible by means of correction templates. We mitigated the threat to satisfaction by using a widely applied model (TAM) [10]. The *hypothesis guessing* threat was mitigated because we did not explain the research questions to the subjects. To weaken the *evaluation apprehension* threat, at the beginning of the experiment, the instructor told the subjects that the experiment was not a test of their abilities. To mitigate the *Author bias* threat, the tasks were extracted from a commercial video game and designed with similar difficulty. Finally, the experiment was affected by the *mono-operation bias* threat because we worked with a single treatment.

External validity. The *domain* threat occurs because the experiment has been conducted in a specific domain, i.e., video-game development. We think that the generalizability of the findings should be undertaken with caution. Other experiments in different domains should be performed to validate our findings.

6 Conclusion

In this work, we present an experiment that compares UML and a DSL when subjects model in the video-game domain. To that extent, we combined objective measures and subjective measures. Our results reveal that the subjects underestimate the potential of their own models. This problem was not discovered by previous works that focus either on objective measures to evaluate modeling performance or on subjective measures to classify modeling styles. Our findings suggest that future evaluations should take into account both objective and subjective measures to better understand modeling languages. Our results are also relevant for teaching modeling and model adoption. If users were able to assess their models correctly, they might leverage their latent potential as programs.

References

1. Agner, L.T.W., Soares, I.W., Stadzisz, P.C., SimãO, J.M.: A Brazilian survey on UML and model-driven practices for embedded software development. Journal of Systems and Software **86**(4), 997–1005 (2013)
2. Akdur, D., Garousi, V., Demirörs, O.: A survey on modeling and model-driven engineering practices in the embedded software industry. Journal of Systems Architecture **91**, 62–82 (2018)
3. Ampatzoglou, A., Stamelos, I.: Software engineering research for computer games: A systematic review. Information and Software Technology **52**(9), 888–901 (2010)
4. Anda, B., Hansen, K., Gullesen, I., Thorsen, H.K.: Experiences from introducing UML-based development in a large safety-critical project. Empirical Software Engineering **11**(4), 555–581 (2006)
5. Bernardino, M., Rodrigues, E.M., Zorzo, A.F.: Performance Testing Modeling: an empirical evaluation of DSL and UML-based approaches. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 1660–1665 (2016)
6. Blasco, D., Font, J., Zamorano, M., Cetina, C.: An evolutionary approach for generating software models: The case of Kromaia in Game Software Engineering. Journal of Systems and Software **171**, 110804 (2021)
7. Budgen, D., Burn, A.J., Brereton, O.P., Kitchenham, B.A., Pretorius, R.: Empirical evidence about the UML: a systematic literature review. Software: Practice and Experience **41**(4), 363–392 (2011)
8. Chaudron, M.R., Heijstek, W., Nugroho, A.: How effective is UML modeling? Software & Systems Modeling **11**(4), 571–580 (2012)
9. Cohen, J.: Statistical power for the social sciences. Hillsdale, NJ: Laurence Erlbaum and Associates (1988)
10. Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Q. **13**(3), 319–340 (Sep 1989)
11. De Lucia, A., Gravino, C., Oliveto, R., Tortora, G.: An experimental comparison of ER and UML class diagrams for data modelling. Empirical Software Engineering **15**(5), 455–492 (2010)
12. Dean, A., Voss, D., Draguljić, D., et al.: Design and analysis of experiments, vol. 1. Springer (1999)
13. Dobing, B., Parsons, J.: How UML is used. Communications of the ACM **49**(5), 109–113 (2006)

14. Domingo, Á., Echeverría, J., Pastor, Ó., Cetina, C.: Evaluating the benefits of model-driven development. In: International Conference on Advanced Information Systems Engineering. pp. 353–367. Springer (2020)
15. Fowler, M.: UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional (2004)
16. Gorschek, T., Tempero, E., Angelis, L.: On the use of software design models in software development practice: An empirical investigation. Journal of Systems and Software **95**, 176–193 (2014)
17. Grossman, M., Aronson, J.E., McCarthy, R.V.: Does UML make the grade? Insights from the software development community. Information and Software Technology **47**(6), 383–397 (2005)
18. Karac, E.I., Turhan, B., Juristo, N.: A Controlled Experiment with Novice Developers on the Impact of Task Description Granularity on Software Quality in Test-Driven Development. IEEE Transactions on Software Engineering (2019)
19. Kosar, T., Gaberc, S., Carver, J.C., Mernik, M.: Program comprehension of domain-specific and general-purpose languages: replication of a family of experiments using integrated development environments. Empirical Software Engineering **23**(5), 2734–2763 (2018)
20. Marko, N.C., Liebel, G., Sauter, D., Lodwich, A., Tichy, M., Leitner, A., Hansson, J.: model-based engineering for embedded systems in practice. Research reports in software engineering and management pp. 1–48 (2014)
21. Moody, D.L.: The method evaluation model: a theoretical model for validating information systems design methods. ECIS 2003 proceedings p. 79 (2003)
22. Mussbacher, G., Amyot, D., Breu, R., Bruel, J.M., Cheng, B.H., Collet, P., Combemale, B., France, R.B., Heldal, R., Hill, J., et al.: The relevance of model-driven engineering thirty years from now. In: International Conference on Model Driven Engineering Languages and Systems. pp. 183–200. Springer (2014)
23. Otero, M.C., Dolado, J.J.: Evaluation of the comprehension of the dynamic modeling in UML. Information and Software Technology **46**(1), 35–53 (2004)
24. Pastor, O., Molina, J.C.: Model-driven architecture in practice: a software production environment based on conceptual modeling. Springer Science & Business Media (2007)
25. Reinhartz-Berger, I., Dori, D.: OPM vs. UML–Experimenting with Comprehension and Construction of Web Application Models. Empirical Software Engineering **10**(1), 57–80 (2005)
26. Staron, M.: Adopting model driven software development in industry–a case study at two companies. In: International Conference on Model Driven Engineering Languages and Systems. pp. 57–72. Springer (2006)
27. Störrle, H.: How are Conceptual Models used in Industrial Software Development? A Descriptive Survey. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. pp. 160–169 (2017)
28. Vegas, S., Apa, C., Juristo, N.: Crossover designs in software engineering experiments: Benefits and perils. IEEE Transactions on Software Engineering **42**(2), 120–135 (2015)
29. West, B.T., Welch, K.B., Galecki, A.T.: Linear mixed models: a practical guide using statistical software. Chapman and Hall/CRC (2014)
30. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
31. Zendler, A., Pfeiffer, T., Eicks, M., Lehner, F.: Experimental comparison of coarse-grained concepts in UML, OML, and TOS. Journal of systems and Software **57**(1), 21–30 (2001)