Do UML Object Diagrams Affect Design Comprehensibility? Results from a Family of Four Controlled Experiments

Marco Torchiano[a], Giuseppe Scanniello[b], Filippo Ricca[c], Gianna Reggio[c], Maurizio Leotta[c]

[a]*Politecnico di Torino, Italy, marco.torchiano@polito.it*
[b]*Università della Basilicata, Italy, giuseppe.scanniello@unibas.it*
[c]*Università di Genova, Italy, {filippo.ricca, gianna.reggio, maurizio.leotta}@unige.it*

Abstract

Objective: The main objective of our study is to assess whether the use of UML (Unified Modeling Language) object diagrams improves comprehensibility of software design when this kind of diagrams is added to UML class diagrams.
Method: We have conducted a family of four controlled experiments. We involved groups of bachelor and master students.
Results: Results suggest that the use of object diagrams does not always introduce significant benefits in terms of design comprehensibility. We found that benefits strongly depend on the experience of participants and their familiarity with UML. More experienced participants achieved better design comprehensibility when provided with both class and object diagrams, while less experienced seemed to be damaged when using class and object diagrams together. Results also showed the absence of substantial variations in the time needed to comprehend UML models, with or without object diagrams.
Implications: Our results suggest that it is important to be aware and take into account experience and UML familiarity before using object diagrams in software modeling.

Keywords: Object Diagram, Family of Experiments, Model Comprehension, UML

1. Introduction

The software engineering community has developed a number of methods and approaches to model software systems. Most of them use the Unified Modeling Language (UML) [1] as the notation to represent structural and behavioral properties of object-oriented software systems. The class diagram is the main building block and it is used to represent object (or conceptual) models during analysis and design [2]. In the analysis phase, class diagrams are used to model entities (or objects of the problem domain) by reporting for each entity some attributes and operations, and relationships among entities. In the design phase, classes are elements of the solution domain.

At class-level we can think of a software system as made up of classes and their associations. To define the behavior of a software, developers write classes that refer to each other. Though, if we observe the program during its execution (i.e., at run-time) we see it as made up of objects linked to each other and interacting; this is the object perspective. Currently the prevailing perspective in the object-oriented community is the class-based. That is, class diagrams are used more frequently than object diagrams.

However, also object diagrams are recognized useful, especially when added to class diagrams. Object diagrams provide a complete or partial view of the structure of a software system at run-time in a given instant [3]. This is even more useful in the analysis phase, where class diagrams added with object diagrams are used to capture the concepts of problem domain of a system, and to highlight relationships among these concepts [2]. Very few evaluations have been conducted in the literature with the goal of assessing object diagram benefits in the software development life cycle [4].

In this paper, we present a family of four controlled experiments to study whether the use of UML object diagrams improves design comprehension of a software system when this kind of diagrams is added to class diagrams. The context of the original (or baseline) experiment [5] is constituted of a group of 17 master students in Computer Science at the Politecnico di Torino. Our first replication was performed at the same university with 17 bachelor students of a Computer Engineering course. Other two experiments are external replications conducted with a group of 24 master students in Computer Science at the University of Basilicata [6] and a group of 66 bachelor students in Computer Science at the University of Genova.

Our study extends [5] and [6]. With respect to these papers, we provide here the following new contributions:

1. Two further replications are presented;
2. Data analysis of individual experiments is presented in a unified way. That is, for each experiment we have adopted the same analysis strategy. Therefore, data from experiments already published (i.e., [5, 6]) have been reanalyzed;
3. Results from our family of experiments are also discussed considering two important context factors (IT experience[1] and familiarity with UML).
4. A more thorough discussion of results is reported;
5. Practical implications for our results are discussed from both the practitioner and the researcher perspectives.

The paper is organized as follows: we discuss in Section 2 a subset of the related literature concerning experiments aimed at assessing the use of UML in comprehension tasks. In Section 3, we present the goal, the research questions, and the experimental objects of our family of experiments. Then, we provide an overview of the original experiment, and describe the design and execution of the three replications. In this section differences among the experiments and analysis method are highlighted as well. We present the family data analysis in Section 4, while we discuss the results of the family of experiments in Section 5. Threats to validity are discussed in Section 6, while remarks and future directions for our research conclude the paper.

2. Related Work

Researchers are looking for ways to improve the comprehension of the various artifacts produced in the software development process. In the early 80s, Woodfield et al. [7] analyzed how different types of modularization approaches and the availability of comments affected the comprehension of programs. In the late 90s, Agarwal et al. [8] compared the comprehension of object-oriented models (tending to focus more on structure) with process-oriented models (tending to emphasize the behavior). The latter kind of models was found to be easier to understand when dealing with complex tasks, whilst no significant difference was observed on simpler tasks.

[1]We use this term since we believe that building IT experience is a long piecemeal process that starts during the university year – if not even earlier – and during such years get systematized.

From the early 2000s, several researchers focused on: (1) analyzing the comprehension of the various UML diagrams (also comparing them with other kind of diagrams), (2) finding ways to improve the comprehensibility of UML diagrams (e.g., adding stereotypes), and (3) analyzing the usage of UML diagrams to comprehend other artifacts, such as requirements or code [9, 10, 11, 12, 13]. This was mainly due to the diffusion of UML both in the academy and in the software industry. In this section, we discuss the literature concerning empirical studies aimed at assessing the comprehensibility of UML models and the use of UML models to comprehend other artifacts. These studies cover the majority of the UML diagrams: Class diagrams, Use Case diagrams, Sequence diagrams, Statechart diagrams[2], Activity diagrams, Collaboration diagrams[3], and Object diagrams. A systematic literature review concerning empirical evaluations on the models and forms used in UML has been conducted by Budgen et al. [4].

Class diagrams. An empirical investigation concerning the comprehensibility of *class diagrams* is proposed by Yusuf et al. [14]. The authors used eye-tracking equipment to assess participants' comprehension in the context of software design problems. Results indicate that more experienced participants prefer to use stereotype information, coloring, and layout to promote the exploration and the navigation of class diagrams.

De Lucia et al. [15] presented results of three controlled experiments involving university students having the goal of comparing the comprehensibility of class diagrams and entity relationship diagrams during maintenance tasks on data models. Results suggested that UML class diagrams are better in terms of comprehensibility but the support given by the two notations during maintenance activities is identical.

Purchase et al. [10] executed an experiment evaluating the comprehension level of five independent "notational variations" for UML class diagrams. These diagrams differed only in the presentation style (e.g., inheritance arcs can be visually presented in overlapped or disjoint way). They found that there is no a best notation; the best performing notation depends on the task for which it is used. Later, the same authors performed a similar experiment but focusing on stylistically different collaboration diagrams [16]. Also in this case results were inconclusive. However, the participants' preferences

[2]In UML 2, they are called State Machine diagrams.
[3]In UML 2, they are called Communication diagrams.

were always in favor of the more concise variants.

A family of controlled experiments to assess the effectiveness of stereotypes in class diagrams to comprehend object-oriented applications in the telecommunication domain is described by Staron et al. [11]. Results suggest that the use of stereotypes significantly helps the participants in improving their comprehension. Similarly, in [17, 18] a family of experiments with bachelor, master, and PhD students is presented to compare the comprehensibility of the Conallen's UML profile with respect to bare UML for modeling Web applications. Differently to [11], stereotypes seem scarcely useful in understanding. The main finding of that family of experiments is that stereotypes reduce the gap between more experienced and less experienced participants.

Use Case diagrams have been empirically investigated by Andrew and Drew [19] with the goal of understanding whether they improve the effectiveness of textual use cases by providing visual clues. The investigation was conducted by providing a group of students only with use cases (control group) or use cases augmented with use case diagrams (treatment group). Results show that students employing both use case diagrams and use cases achieve a significantly higher level of understanding. In another paper [20], the comprehensibility of requirements models expressed in two visual modeling notations, Use Case, which is a scenario-base language, and Tropos, which exploits goal-oriented modeling, is compared. This experimental evaluation has been conducted within a family of controlled experiments involving 79 university students overall. The experimental results showed that Tropos models seem to be more comprehensible, although more time consuming than Use Case models.

Sequence diagrams have been considered in several empirical studies. For instance, Xie et al. [21] focused on sequence diagrams used to model multi-threaded concurrency. In particular, they proposed a variation of this kind of diagram called synchronization adorned UML (saUML) sequence diagram. Xie et al. evaluated saUML sequence diagrams by mean of an empirical experiment. A statistically significant benefit from adopting saUML sequence diagrams was found, thus the authors concluded that this variant of sequence diagrams improves the comprehension of concurrent programs.

A family of three experiments, involving undergraduate computer science students, carried out to investigate the influence of using stereotypes in UML sequence diagrams is presented by Cruz-Lemus et al. [22]. In particular, the authors studied the comprehensibility of UML sequence diagrams with and without stereotypes from three different perspectives: semantic, retention, and transfer. The authors found that the use of stereotypes improves comprehensibility, particularly for participants not familiar with the application domain of the object understudy.

Gravino et al. [23] present a controlled experiment executed with bachelor students to assess whether comprehension of software requirements is influenced by the use of dynamic models abstracted by employing sequence diagrams. The most important result was: the difference in comprehension of system requirements is not statistically significant when using or not dynamic models. On the contrary, results from an external replication conducted with more experienced participants (i.e., master students) showed that the use of dynamic models facilitates comprehension of requirements. Results were confirmed in three subsequent replications conducted in Italy and Spain with professionals and master/PhD students [9].

Finally, Glezer et al. [24] compare two types of interaction diagrams (sequence and collaboration) in two application domains: management information systems and real-time systems. Results suggest that collaboration diagrams are easier to comprehend than sequence diagrams in real-time systems, but this is not true in management information systems; in these last systems there is no difference in comprehension of the two diagram types.

Statechart diagrams have been extensively empirically investigated, too. For example, Cruz-Lemus et al. [25] studied the impact of composite states (which allow to group related states) on understandability of UML statechart diagrams. They carried out three empirical studies, consisting of five experiments in total using relatively small statechart diagrams as experimental objects while, as participants, they employed undergraduate and graduate students of Computer Science at several universities, along with a number of professionals with an average of two years of experience in UML modeling. The authors stated that the use of composite states improves understandability and efficiency when statechart diagrams are consulted/analyzed. Nevertheless, participants experience with statechart diagrams was considered essential to gain the improved understandability.

Activity diagrams have been empirically investigated in comprehensibility context. For this kind of diagrams, we selected two papers. The first work [26] reports on two controlled experiments aimed at comparing UML activity diagrams and the event-driven process chains (EPCs). The main objective of that comparison concerned the model understandability from the perspective of both requirements engineers and customers. In the case of requirements engineers, better performances have been obtained when participants use activity diagrams. On the other hand, the customers do not get significant differences in terms of business process understandability, when they use both the business process visual notations.

Reggio et al. [27] presented the results from two experiments with bachelor and master students. These experiments have been conducted to assess whether the level of formality/precision in business process modeling [28], based on UML activity diagrams, influences two aspects of comprehensibility: correctness of understanding and task completion time. Considered styles were: a precise style (with specific rules and imposed constraints) and an ultra-light style (no rules, no imposed constraints). Results indicate that: the participants achieved a significantly better comprehension level with the precise style, the used style did not have any significant impact on the effort and more experienced participants benefited more from the precise style than the ultra-light style.

Object diagrams and inexperienced students have been considered in the experiment by Thomas et al. [29]. The authors involved a group of first year programming students attempting multiple-choice code tracing questions[4] on an object-oriented program. Students were provided (or not) with partially completed object diagrams. The goal of the experiment was testing the experimental group (source code provided with objects diagrams) against the control group (source code provided without objects diagrams). The conjecture is that the students belonging to the experimental group would do better than the control group since they had at least some help. This turned out not to be the case since results were inconclusive. Results obtained by Thomas et al. [29] reinforce our findings: the use of object diagrams does not always introduce significant benefits in terms of comprehension, but it depends on IT experience and familiarity with UML. In addition, results of two out of four conducted experiments were consistent with those by Thomas et al. [29]: providing inexperienced students with object diagrams do not improve their understanding. Although this study might be considered similar to that we present in this paper, there are two remarkable differences: type of empirical study (controlled experiment vs. family of experiments involving students with different IT experience and different familiarity levels with UML) and type of comprehension questions (code tracing questions vs. more

[4]Code tracing questions require students to demonstrate their understanding of code's execution, or describing what the code does in general.

general questions concerning related class diagram and application domain).

3. The Family

In this section, we present the goal of our family of experiments, research questions, and used experimental objects. Then, we present an overview of both the baseline experiment and the three replications. Finally, we summarize differences among experiments in our family and data analysis procedure adopted to compare results from individual experiments.

The baseline experiment (also *PoliTo1*, from here on) [5] was carried out at the Politecnico di Torino in 2003. It was first internally replicated in 2004 (*PoliTo2*, from here on) and later it was externally replicated two times: at the University of Basilicata in 2010 (*UniBas1*, from here on) [6] and at the University of Genova in 2011 (*UniGe1*, from here on). External replications allowed us to increase external validity. For replication purposes, we made available on the web[5] an experimental package and the raw data of all the experiments. To design and execute the experiments, we followed the guidelines proposed in [30, 31].

3.1. Goal and Research Questions

On the basis of the Goal Question Metric (GQM) template [32], the goal of our family of experiments can be defined as follows:

Goal: *Analyze* the use of UML object diagrams *with the purpose* of assessing whether they improve design comprehensibility of software design when these diagrams are added to UML class diagrams. The *quality focus* is to ensure high comprehensibility, taking the *perspective* of both Researchers, evaluating how effective are object diagrams, and Project managers, evaluating the possibility of adopting object diagrams in their organizations, *in the context* of students having different IT experience and different familiarity with UML.

In our family of experiments, we formulated and investigated the following two research questions:

RQ1 Does the *effectiveness* to comprehend software design vary when object diagrams are used in addition to class diagrams?

[5]www2.unibas.it/gscanniello/ObjectDiagrams/

RQ2 Does the *effort*[6] required to complete a comprehension task vary when object diagrams are used in addition to class diagrams?

3.2. Experimental Objects

We used the following four experimental objects in all the experiments:

FS. The File System Manager handles folders, files, and links. Folders can contain other elements (e.g., files), while links refer to other elements in the file system (e.g., folders). Both the used class diagram (up) and object diagram (down) are shown in Figure 1;

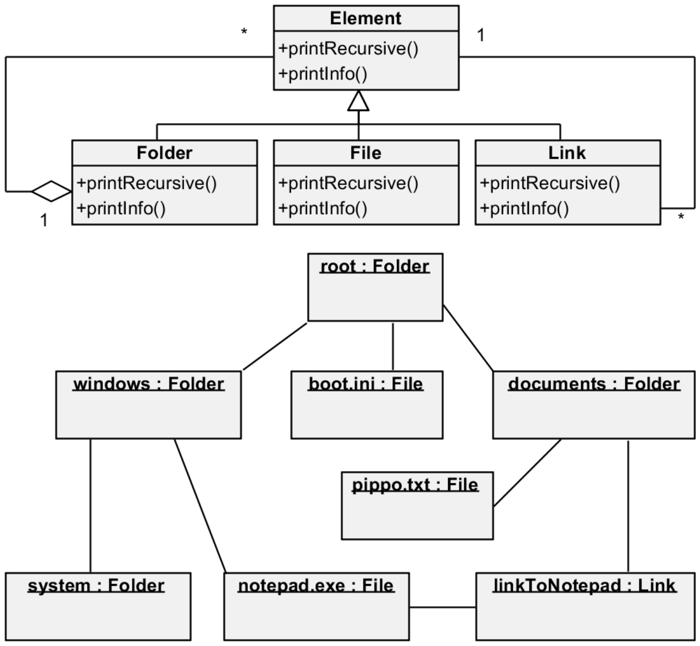

Figure 1: The class and object diagrams of FS.

R. The Roads system handles maps made up of cities connected by means of roads. Each road starts and ends in a city. Furthermore, a road is characterized by a length. Both the used class diagram (left) and object diagram (right) are shown in Figure 2;

[6]We consider the time as an approximation for effort. This is almost customary in literature (e.g., [12]) and it is compliant with the ISO/IEC 9126 standard, namely effort is the productive time associated with a specific project task.

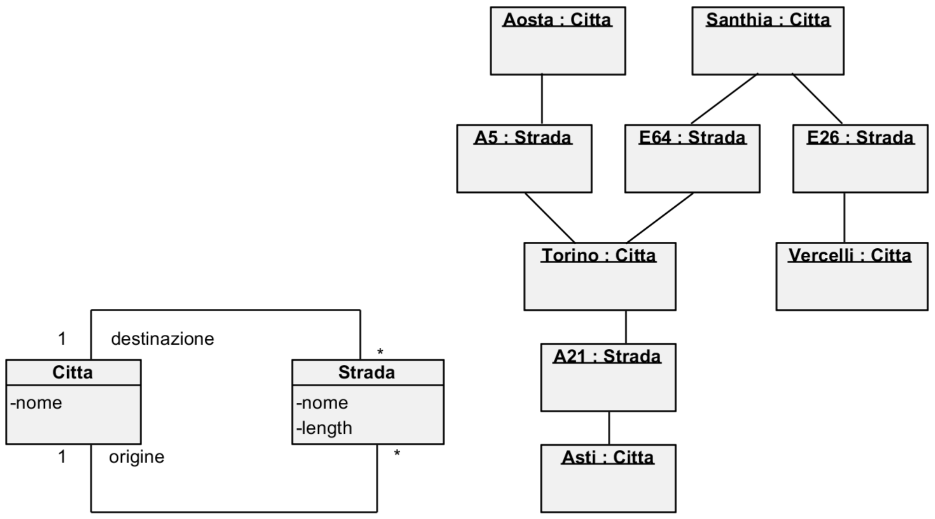

Figure 2: The class and object diagrams of R.

T. Train is a system to manage timetables, trains, and paths. Both the used class diagram (up) and object diagram (down) are shown in Figure 3;

C. The Catalogue system collects category of items (e.g., cars) and items (e.g., car models) based on a set of features (e.g., number of doors). Both the used class diagram (up) and object diagram (down) are shown in Figure 4;

Some details on the experimental objects are shown in Table 1. We used four experimental objects to diversify the application domains so as to reduce external validity threats (see Section 6).

To observe a possible contribution of object diagrams, we selected UML class diagrams having a significant structure (i.e., containing classes with

	Class diagram		**Object diagram**	
Object	**Classes**	**Relations**	**Instances**	**Links**
FS	4	5	8	8
R	2	2	9	8
T	4	5	15	22
C	5	6	6	7

Table 1: Characteristics of experimental objects used in the family of experiments.

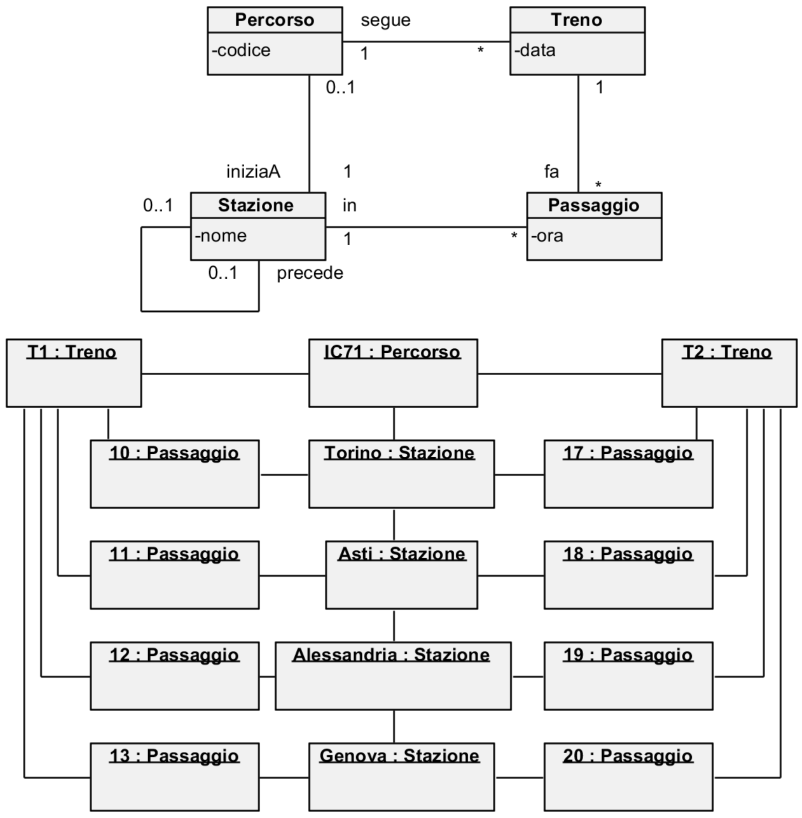

Figure 3: The class and object diagrams of T.

several associations). In particular, the class diagram of *File System* (FS) has a tree-like structure that is achieved by means of an indirect self-loop association (i.e., an instance of the design pattern composite [33]). The class diagram of *Roads* (R) contains two classes and two associations forming a loop that allows representing a graph, while the class diagram of *Trains* (T) has four classes forming a cycle and an additional self-loop association on class Station. The *Trains* system presents a type-instance duality: Train-Path and Station describe a “type”, while Train and Passage can be seen as “instances” of this type. Finally, the class diagram of *Catalogue* (C) contains cycles (but no self-loop association) and presents a type-instance duality. The used models of the experimental objects can be downloaded from the home page of our family of experiments.

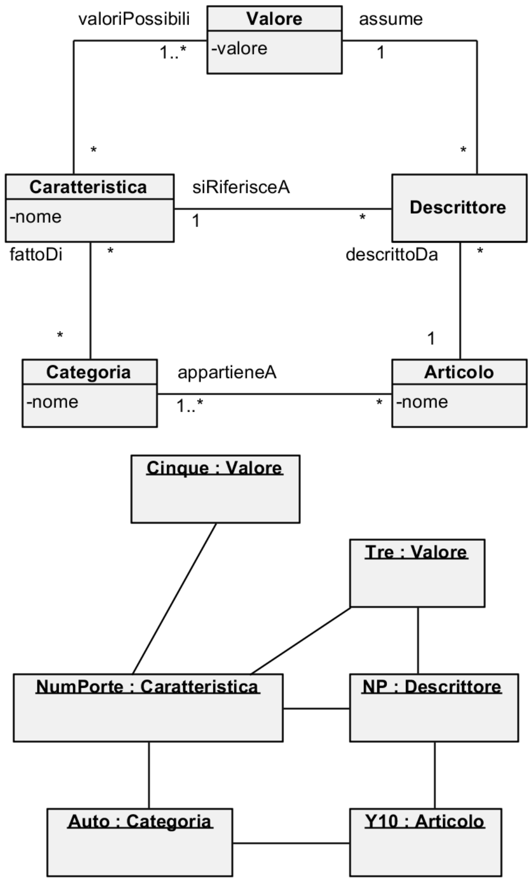

Figure 4: The class and object diagrams of C.

3.3. PoliTo1

17 master students carried out PoliTo1 [5]. They were enrolled at the first year of the Master program in Computer Science. Some of them were, or had been, industry professionals. The experiment was a class exercise of a basic course in software development including a quick introduction to UML and object-oriented programming. These students had no previous experience with UML. In their career, they attended only introductory computer science courses.

Participants were required to perform some comprehension tasks on the experimental objects alternating the following treatments:

– the system is documented using a textual description supplemented by a class diagram only (no object diagram is provided);

	Groups	
Tasks	**Group 0**	**Group 1**
Task 1	FS+	FS-
Task 2	R-	R+
Task 3	T+	T-
Task 4	C-	C+

Table 2: PoliTo1 experiment design. Class diagram augmented with object diagram (+), only class diagram (-)

+ the system is documented using a textual description supplemented by both a class diagram and an object diagram.

The experiment used a factorial design with two groups, one treatment, and four tasks. Each task was performed on a single experimental object according to the schema shown in Table 2. Participants were randomly assigned to the groups Group 0 and Group 1.

The original experiment was designed to investigate RQ1 only. For this reason only the comprehension level of participants was measured. The comprehension achieved on each experimental object was assessed through a questionnaire. It included four multiple-choice questions with only one correct answer (i.e., closed-ended questions) for each experimental object. Each question could be answered examining provided UML models (depending on the administered treatment: class diagram augmented with object diagram or class diagram only). That is, we designed the experiment so as to the class diagrams alone were sufficient to answer the questions of comprehension questionnaire.

An example of comprehension question for FS is: "*Which object types can belong to a Folder object?*". To answer this question is sufficient looking at the aggregation present in the class diagram (see on the top of Figure 1) or to consider also the example provided by object diagram (see on the bottom of Figure 1). The correct answer for this question is composed by the following items: Folder, File, and Link.

The comprehension level achieved by a participant has been measured counting the number of correct answers in the comprehension questionnaire (i.e., *score*). For each experimental object, the comprehension level achieved by a given participant in the experiment ranges in between 0 (null comprehension) and 4 (perfect comprehension).

3.3.1. Summary of Results for PoliTo1

We give here a summary of PoliTo1 results. The interested reader can found further details on this experiment in [5]. Data analysis was performed considering experimental objects alone and together. Results on each object alone revealed a significant statistical difference in favor of object diagrams on FS and T. Given the preliminary nature of PoliTo1, the author accepted 17% of committing Type-I-error instead of 5% [31]. Such an alpha level has been chosen to have the statistical power of the Mann-Whitney test greater than 75%. In all other experiments in our family, we decided to accept a probability of 5% of committing Type-I-error. The analysis of the magnitude of the observed differences, measured using the Cohen's "*d*" standardized difference between two groups [34], revealed that the effect size[7] was medium for FS and T and negligible for R and C. A significant positive effect of object diagrams was observed when considering all the experimental objects together. The effect size was small.

3.4. PoliTo2

Participants in PoliTo2 were 17 bachelor students of a Computer Engineering degree. The experiment was a class exercise of an Object-Oriented Programming course, which included an introduction to UML. Before attending this course, students had no previous experience with UML. In particular, they had only attended an introductory programming course and an Algorithms and Data Structures course. Therefore, they had a limited familiarity with UML and a limited experience in software development. Results from this experiment have never been published.

Also in this case the experiment was designed to investigate only RQ1. To reduce the time required to conduct the replication (only one hour and half was available in total) and to adapt the experiment to the limited experience of students, only FS and T were used. We adopted a within-participants counterbalanced design with two tasks and four groups summarized in Table 3. We randomly assigned participants to the four groups shown in table.

3.5. UniBas1

We conducted UniBas1 in 2010 with 24 students of the Master program in Computer Science at the University of Basilicata (Italy) [6]. The experiment

[7]Typically, the effect size is considered negligible for $d < 0.2$, small for $0.2 \leq d < 0.5$, medium for $0.5 \leq d < 0.8$, and large for $d \geq 0.8$.

	Groups			
Task	**Group 0**	**Group 1**	**Group 2**	**Group 3**
Task 1	FS-	FS+	T-	T+
Task 2	T+	T-	FS+	FS-

Table 3: PoliTo2 experiment design. Class diagram augmented with object diagram (+), only class diagram (-)

	Groups			
Task	**Group 0**	**Group 1**	**Group 2**	**Group 3**
Task 1	A1+	A1-	A2+	A2-
Task 2	A2-	A2+	A1-	A1+

Table 4: UniBas1 experiment design. Class diagram augmented with object diagram (+), only class diagram (-)

represented an optional activity of an advanced Software Engineering course.

Before the execution of UniBas1, participants have already passed a basic Software Engineering course, where they learned UML. Some of them were, or had been, industry professionals. Thus, we can state that participants in UniBas1 had a reasonable level of technical maturity and knowledge of UML, requirements engineering, and object-oriented software development.

In this replication, experimental objects were paired in *assignments* as follows:

A1. the pair of objects FS and T;

A2. the pair of objects R and C.

We adopted the within-participants counterbalanced design shown in Table 4. This design ensures that each participant work on two tasks (each consisting of an assignment), experimenting alternatively class and object diagrams together (+) or class diagrams alone (-). Participants performed the two tasks without time limit. Participants could have a break between the two tasks. We opted for within-participants counterbalanced design because it allows reducing possible carry-over effects.[8]

[8]If a participant is tested first under the condition X and then under the condition Y, he/she could potentially exhibit better or worse performances (with respect to a given dependent variable) under the second condition.

We equally distributed high and low ability participants among the four groups (i.e., Group 0, Group 1, Group 2, and Group 3) [6]. This was possible because we asked participants to fill out a pre-questionnaire. In particular, we asked participants their GPA (Grade Point Average), and used this information to assign them to the groups. We equally distributed participants with a GPA less than or equal to 24 (low ability participants) in the four groups in Table 4. This design choice was applied in previous studies (e.g., [9]).

The scope of UniBas1 is broader than PoliTo1 (i.e., the original experiment). In fact, we also measured comprehension effort to answer RQ2. Therefore, the considered dependent variables are: *comprehension level* and *comprehension effort*.

Similarly to PoliTo1, the comprehension level has been assessed by employing a comprehension questionnaire. However, we decided to convert the questions in the comprehension questionnaire in open-ended questions. This allowed to mitigate possible threats to the validity of results. That is, this decreases the possibility participants gave correct answers by chance. Therefore, the comprehension level has been measured using an information retrieval based approach [17]. In particular, correctness of provided answers has been measured using the *precision* measure. On the contrary, for estimating the completeness of the answers, we exploited the *recall* measure. To get a single value representing a balance between correctness and completeness of the answers, we used the F-measure (i.e., the harmonic mean between precision and recall). A single value for the comprehension level is obtained computing the overall average of the F-Measure values of all the questions. This mean assumes values ranging from 0 to 1. A value close to 0 indicates a null comprehension of the design, whilst a value close to 1 means a complete comprehension. Further details on that dependent variable *comprehension level* can be found in [6].

The dependent variable *comprehension effort* measures the time to accomplish a task. The time was expressed in minutes and was directly recorded by participants by writing down their start and stop times on comprehension questionnaires.

A post-experiment questionnaire was administered at the end of the experiment with the goal of gaining insights about participants' behavior during the experiment. That is, we used a qualitative approach to explain the quantitative results from experiments. To design this questionnaire, we used standard approaches and scales [35]. This allowed us to reduce possible experimenters' biases [31].

3.5.1. Summary of Results for UniBas1

Table 5 summarizes results in terms of comprehension level (both significance and effect size) of UniBas1. For comparison purposes, this table also shows results obtained in the original experiment. The interested reader can found further details in [6].

	Significance		**Effect size**	
Object	**PoliTo1**	**UniBas1**	**PoliTo1**	**UniBas1**
All	yes	yes	small	large
FS	yes	yes	medium	large
T	yes	yes	medium	large
R	no	yes	negligible	large
C	no	yes	negligible	large

Table 5: Comparison between PoliTo1 and UniBas1 in terms of comprehension level

Results confirm (see row "All") and strengthen findings of baseline experiment, thus increasing our confidence in benefits deriving from the use of object diagrams. Complementing class diagrams with object diagrams improves comprehension of circa 35%.

Regarding the comprehension effort, the experiment did not reveal any significant difference. This result suggests that the provided additional information (i.e., object diagrams) does not require addition effort to complete a comprehension task.

Finally, the analysis of the answers to the post-questionnaires suggests the usefulness of object diagrams. In addition, participants found object diagrams useful to understand the cardinality of associations.

3.6. UniGe1

Participants in UniGe1 were 66 students of the Bachelor program in Computer Science. These students were enrolled in a Software Engineering course of the third year, where they learned UML. As a mandatory activity of this course, students were grouped in teams and allocated on projects to develop a software system using specifications based on UML. As for UniBas1, the experiment was conducted as part of laboratory exercises carried out within the course in which the experiment was conducted. Participants had knowledge of object-oriented programming and database systems modeling. To conduct this replication, we used the same design as UniBas1.

3.7. Differences Among the Experiments

We introduced some differences in replicated experiments. Some of these differences have been intentionally introduced to improve material and/or data analysis and then to increase the validity of results. Other variations were added because of time constraints (e.g., the laboratory was available only for two hours). We summarize differences as follows:

Comprehension questionnaire. Before executing the three replications, we removed some minimal sources of possible confusion found in the material of PoliTo1. Furthermore, in UniBas1 and UniGe1, we modified questionnaires by turning closed questions into open ones. The rationale for this modification relies on the fact that open questions should reduce participants guess and hence decrease the possibility to give correct answers by chance.

Dependent variables. Apart from comprehension, we added effort as dependent variable in UniBas1 and UniGe1. This variable was added to further investigate the effect of object diagrams.

Experiment design. For PoliTo2, UniBas1 and UniGe1, we used a within-participants counterbalanced experimental design. On the contrary, a factorial design with two groups, one treatment, and four objects was used in PoliTo1. We added this modification to better analyze the effect of the main factor (i.e., the presence or the absence of object diagrams to execute a comprehension task) and co-factors (e.g., task). Other modifications have been introduced as a natural consequence of having used a different design. Another modification has been introduced in UniBas1. That is, a break between two tasks was provided. This modification was introduced to reduce fatigue on the main factor understudy. As for UniGe1, we had time constraints and then participants did not have a break between tasks.

Group composition. For UniBas1, we used the information from a pre-questionnaire to equally distribute high- and low-ability participants among the groups shown in Table 4. In the baseline experiment, and in the PoliTo2 and UniGe1 replications, participants were randomly assigned to groups.

Post-experiment survey questionnaire. For PoliTo2 and UniBas1, a post-experiment questionnaire was employed.

	PoliTo1	**PoliTo2**	**UniBas1**	**UniGe1**
Number of participants	17	17	24	66
Kind of students	Master	Bachelor	Master	Bachelor
Geographic distribution	Torino	Torino	Potenza	Genova
Research questions	RQ1 only	RQ1 only	RQ1 and RQ2	RQ1 and RQ2
Comprehension questionnaire	Closed answers	Closed answers	Open answers	Open answers
Design	Factorial 2	Counterbalanced	Counterbalanced	Counterbalanced
Tasks	4	2	2	2
Comprehension level	Score	Score	F-measure	F-measure
Comprehension effort	No	No	Yes	Yes
Time limit	Yes	Yes	No	Yes
Break between the tasks	No	No	Yes	No
Group composition	Randomized	Randomized	Blocking factor (Ability)	Randomized
Post-experiment questionnaire	No	Yes	Yes	No
Participants' IT experience	High	Low	High	Low
Familiarity with UML	Low	Low	High	High
Empirical strategy	Quantitative	Quantitative	Quantitative and Qualitative	Quantitative

Table 6: Key features of the four experiments in our family

Data analysis. Owing to the new dependent variable introduced (i.e., comprehension effort), we considered a new null hypothesis in UniBas1 and in UniGe1. Moreover, in all the replications of our family we accepted the standard probability of 5% of committing Type-I-error instead of 17% as done in the original experiment.

Participants' IT experience. We measured participants' experience as their level of education, i.e., either bachelor or master. Master students were considered to have high IT experience, while bachelor students low.

Familiarity with UML. The familiarity of participants with UML has been measured as high or low. It has been estimated by the lecturer on the basis of UML contents provided to students both in the course where experiment took place and in previous university courses.

We summarize main differences among experiments in Table 6.

3.8. Family Analysis Procedure

Since the four experiments have been conducted in different settings and with different number of tasks, we could not combine results and analyze data as originally gathered. Moreover, the measure to quantify comprehension level in the two experiments conducted at Politecnico di Torino (PoliTo1 and PoliTo2) is different from the measure used in UniGe1 and UniBas1 (score vs. F-measure). We devised a transformation of measures that allowed for a direct comparison of results from our experiments. The idea is to discretize comprehension level (computed as F-measure in the UniBas1 and UniGe1) into a dichotomous variable indicating whether an answer is correct or not (i.e., analogous to the variable used in PoliTo1 and PoliTo2). The threshold we adopted for such discretization is the most conservative one, i.e., correct $\Leftrightarrow$ F-measure = 1. This transformation is similar to that used in [36, 37].

After this transformation, we report summary statistics of four experiments in terms of absolute number of correct answers per task (score) with and without object diagrams. Since the distributions of the scores are not normal, we use the median and the median absolute deviation (MAD). We also provided results of the Mann-Whitney test, which were consistently applied in all experiments of our family.

Since the number of questions per task is different among four experiments (namely four in PoliTo2 and eight in the others), the score is not a suitable measure for a comparison among the experiments in our family. Therefore, we decided to build, for each experiment, a contingency matrix of correct/wrong answers vs. the presence/absence of object diagrams.

The contingency table allows us to look at the benefits achievable by object diagrams in terms of odds ratio, which is a measure of effect size that can be used for dichotomous categorical data. An odds ratio indicates the likelihood of occurrence of an event as opposed to non occurrence. Odds ratio is defined as the ratio of the odds of an event occurring in one group (e.g., experimental group) to the odds of it occurring in another group (e.g., control group), or to a sample-based estimate of that ratio. If the probabilities of the event in each of groups are indicated as p (experimental group) and q (control group), then the odds ratio is precisely defined as:

$$OR = \frac{p/(1-p)}{q/(1-q)}$$

An odds ratio of 1 indicates that the condition or event under study is equally likely in both groups. An odds ratio greater than 1 indicates that the condition or event is more likely in first group. Finally, an odds ratio less than 1 indicates that condition or event is less likely in first group.

Table 7: Summary statistics of raw score for the four experiments

	NO_OD		OD		MW
Exp	**median**	**MAD**	**median**	**MAD**	**p-value**
PoliTo2	3.00	1.48	3.00	1.48	0.77
UniGe1	5.73	1.58	5.58	0.96	0.77
PoliTo1	3.00	1.48	3.00	1.48	0.09
UniBas1	4.50	1.68	6.50	0.74	$<$ **0.01**

Moreover, on the basis of the contingency tables we apply the Fisher exact test, which is well suited for small samples 2×2 tables [38]. The test allows to check for the existence of a significant difference between two treatments in term of correctness.

We also studied how contexts influenced benefits achievable with object diagrams. The two context factors considered are IT experience and familiarity with UML. Both are ordinal measures that can assume High and Low as the values, respectively. To identify a possible correlation between the two context factors and the experiment outcome we fit a linear model. Given the nature of factors and the lack of other empirical evidence, we opted for simplest model.

4. Family Data Analysis

In this section, we present the results of a data analysis conducted on our family of experiments according to defined research questions.

4.1. RQ1 - Comprehension Level

Summary statistics on comprehension level are reported in Table 7. This table reports median and median absolute deviation (MAD) for both treatments. Table 7 also reports p-values the Mann-Whitney test returned. We can observe a statistical significant difference only in UniBas1.

Figure 5 shows the boxplots for comprehension level grouped by treatment and experiment. The number of participants is proportional to the widths of boxes. Boxes can be only pairwise compared because experiments differ in both number of tasks and number of questions per task.

As mentioned in Section 3.8, we use the odds ratio of a correct answers to make comparable outcomes of four experiments. Table 8 reports odds ratio

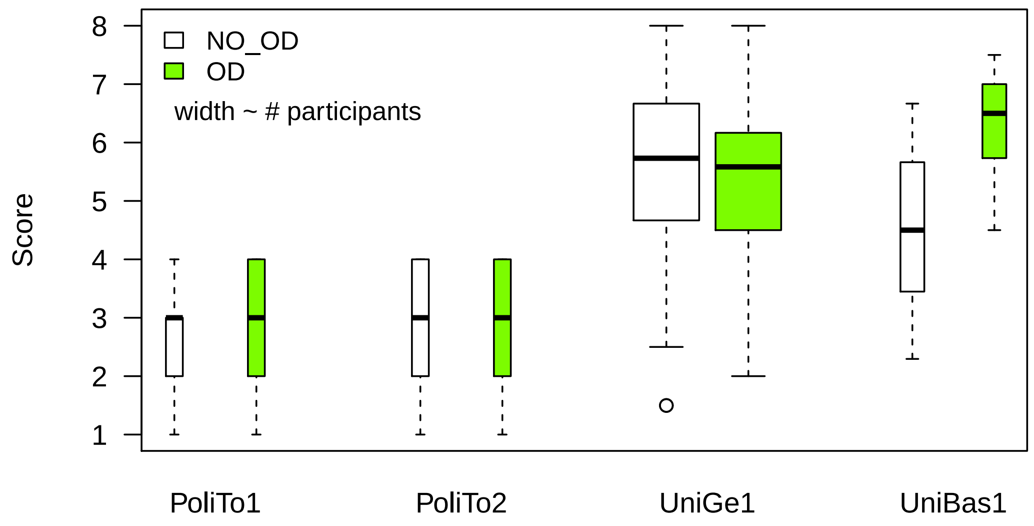

Figure 5: Score in the four experiments.

Table 8: Odds ratio estimate with 95% CI in the four experiments

	Odds ratio			Fisher
Exp	est.	95% CI		p-value
PoliTo2	0.73	0.31	1.71	0.55
UniGe1	1.08	0.84	1.40	0.57
PoliTo1	1.39	0.79	2.47	0.28
UniBas1	1.76	1.15	2.70	< **0.01**

(OR) estimates and 95% confidence interval[9], with p-values of the Fisher test. Based on either the odds ratio confidence interval or the Fisher test, we can reject first hypothesis only for UniBas1 with a large effect size (OR=1.76), while we cannot reject it for other three experiments. This result is consistent with the analysis summarized in Table 7.

A graphical representation of odds ratios and relative confidence intervals is shown in Figure 6. Both estimates and intervals can be directly compared. We can graphically infer the significance of effect observed in UniBas1, and the lack of statistical significance in the other experiments (the lower extreme of CI overcomes the straight line OR=1 only for UniBas1).

[9]The width of the confidence interval gives us some idea about how uncertain we are about unknown parameter (in our case odds ratio estimate)

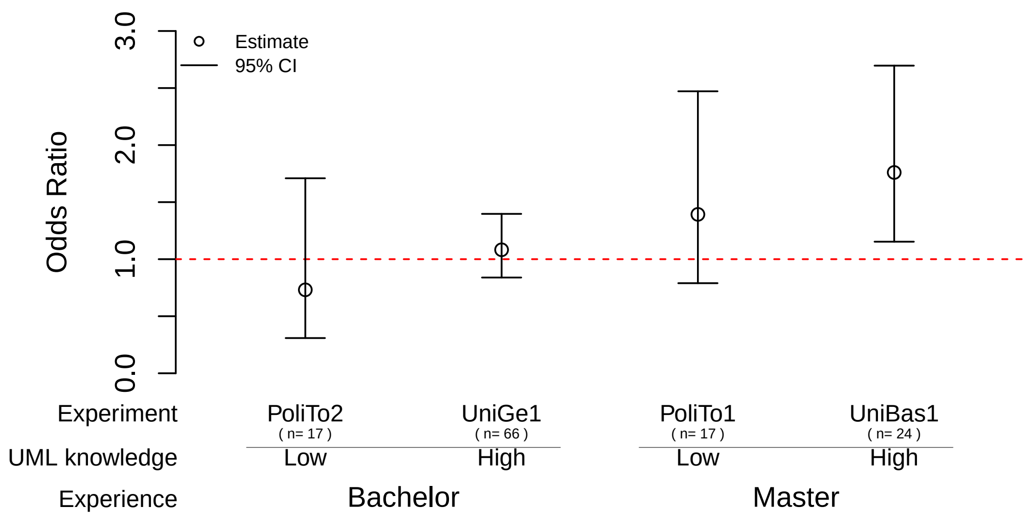

Figure 6: Odds ratio estimate with 95% CI in the four experiments.

The effect of context factors – IT experience and UML familiarity – can be estimated by means of a linear model. Fitting a linear model on four estimates vs. IT experience and UML familiarity levels yields the following equation:

$$OR = 0.73 + 0.67 \cdot MasterITExperience + 0.36 \cdot HighUMLFamiliarity \quad (1)$$

Where *MasterITExperience* is 1 for master students and 0 for bachelor students. *HighUMLFamiliarity* is 1 for participants with a high familiarity with UML and 0 otherwise.

All the three coefficients are statistically significant as well as the overall model (p-value = 0.0111). The goodness of fit is very high ($R^2 > 99\%$)[10].

4.2. RQ2 - Comprehension Effort

As far as effort is concerned, only UniBas1 and UniGe1 collected timing information. Values are summarized in Figure 7. A statistically significant difference was found neither in UniBas1 (p=0.22) nor in UniGe1 (p=0.99).

[10] R^2 measures how well a regression approximates the real data points; R^2=100% indicates that the regression perfectly fits the data.

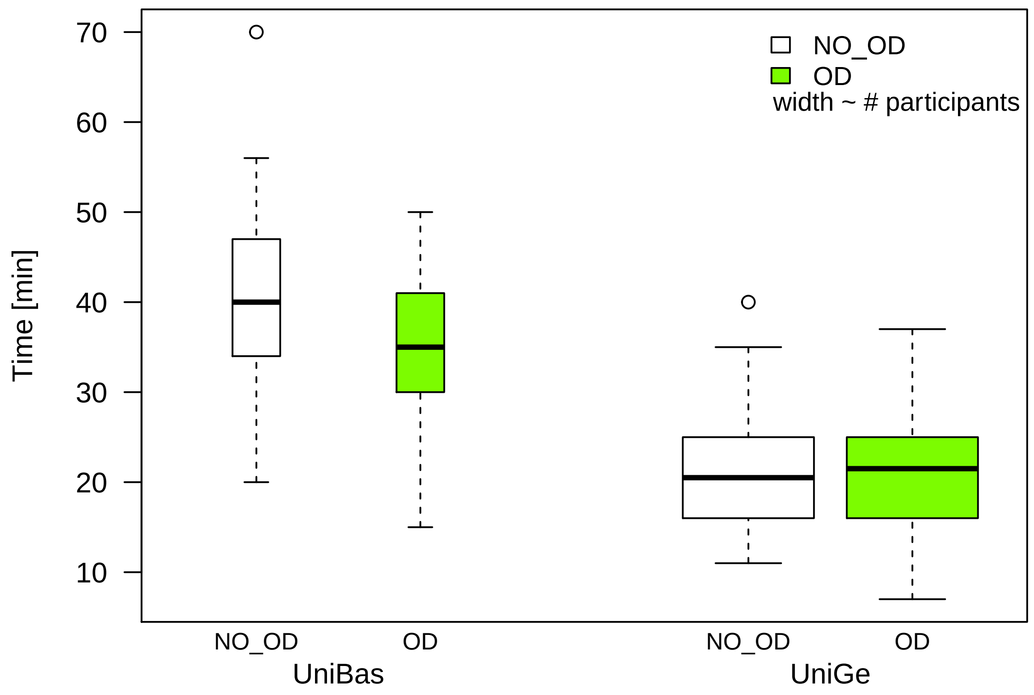

Figure 7: Time to complete the task in UniBas1 and UniGe1.

5. Discussion

Running a family of experiments rather than an individual experiment provides us with more evidence about external validity – including generalizability – of results [39, 40]. The same hypotheses were tested in three distinct contexts (Politecnico di Torino, University of Genova, and University of Basilicata) employing two different profiles of participants (master and bachelor students) having different familiarity with UML.

The first result that emerges from our family of experiments is that the use of object diagrams does not always introduce significant benefits in terms of design comprehension. Our result partially contrasts a general belief that UML object diagrams are useful to increase the comprehension of class diagrams in the modeling of object-oriented software systems. For example, Fowler wrote: *Object diagrams are useful for showing examples of objects connected together. In many situations, you can define a structure precisely with a class diagram, but the structure is still difficult to understand. In these* *situations, a couple of object diagram examples can make all the difference* [3].

We found that the variability of our results strongly depends on two co-factors: IT experience and UML familiarity. These co-factors play an important role in our family of experiments. Students having a higher IT experience (master) and high UML familiarity obtained clear benefits. On the contrary, less experienced participants (i.e., bachelor students) having low UML familiarity obtained no advantage from the usage of object diagrams (this outcome is consistent with those obtained by Thomas et al. [29]). This is quantitatively shown by the odds ratio that provides a measure of the effect size. More in detail, master students having high UML familiarity improve their comprehension level of UML models of circa two times (OR=1.76), when object diagrams are provided. The improvement is reduced when the familiarity with UML decreases, OR = 1.39 in the PoliTo1 experiment where the students had a low familiarity with UML, and collapse with less IT experienced students (OR=1.08 in UniGe1 and OR=0.73 in PoliTo2). The relationship between IT experience and UML familiarity and the magnitude of benefits in terms of comprehension of UML models is highlighted by the linear model (equation 1). From this model, we observe that the coefficient for IT experience is roughly two times larger than the coefficient for UML familiarity, indicating a larger importance of IT experience in determining the magnitude of benefits achievable from the presence of object diagrams.

The fact that students having more IT experience and high UML familiarity obtained the highest benefits in the usage of object diagrams is, in some sense, counter-intuitive. Our results contrast the common belief that specific examples (in our case object diagrams) are more useful than generalizations (in our case class diagrams) for novices while examples are less useful for experienced participants. In our family of experiments, we obtained exactly the opposite. This result probably depends on the capability of participants to integrate different sources of knowledge, namely class diagrams and objects diagrams. Some of them are able to quickly find associations between different artifacts and for them object diagrams are useful, while others find this kind of notation more difficult and in this case having or not the objects diagrams does not make difference. We observed a similar phenomenon in the study by Reggio et al. [27], where a precise style and an ultra-light style for modeling business processes were compared. Also in that case, more experienced participants (master students) benefited more from the precise style, in which a combination of UML diagrams has been used to model a business process, than less experienced participants. Similarly Abrahão et al. [9]

observed that in the case of high ability and more experienced participants the use of sequence diagrams, in combination with other UML notations, improves functional requirements comprehension.

As for comprehension effort, the results confirmed the absence of substantial variations in the time needed to perform a comprehension task, whatever is the treatment applied.

5.1. Implications

We adopted a perspective-based approach to judge practical implications for our family of experiments from the *practitioner/consultant* (simply practitioner, here on) and the *researcher* perspectives. To this end, we exploited the reading method inspired by perspective-based and checklist-based reviews suggested Kitchenham et al. [41]. The practical implications suggested by our results can be summarized as follows:

- Comprehension of software design improves when class diagrams and object diagrams are provided together only in certain cases. The improvement depends on the IT experience and the familiarity with UML of participants. This outcome is relevant for practitioner. We should expect an improvement in comprehension only for software engineers with a little of IT experience and good familiarity with UML. This outcome is also relevant for the researcher. Understanding the reason of such result from a cognitive point of view is a challenge and our results together with those previously published in the literature (e.g., [9],[27]) pose the basis for future work.

- The use of object diagrams could reduce the number of defects originated from the model comprehension. In addition, the use of such a kind of diagram could also positively impact communication among stakeholders. This is true only in case stakeholders have a given experience and familiarity with UML. This implication is clearly relevant for the practitioner.

- Using different layouts for a diagram could lead to different comprehension level of software design. This aspect is relevant for the practitioner, who could be interested in understanding the best way for creating UML class and object diagrams. Even the researcher could be interested in such a result, so investigating why a different layout should affect comprehension of software design. Although our empirical work has not been specifically designed for pursuing these concerns, our outcomes pose the base for future work. That is, IT experience and UML familiarity have to be taken into account in designing future empirical studies on the effect of layouts on diagram comprehensibility.

- The use of object diagrams induces no additional time burden, while performing a comprehension task. This result is relevant for practitioners/consultants even if this opens an issue that deserves future dedicated empirical investigations: does the use of object diagrams impact on the time to create/maintain UML models? Researchers could be interested in such a kind of future research direction.

- The study focused on UML models in domains in which participants were familiar with. We cannot claim that, in less familiar domains, the gap in comprehensibility between two treatments (class diagrams plus object diagrams with respect to class diagrams alone) would increase or decrease. However, based on the experience we gained in the family of experiments presented in this paper, we might postulate that the presence of object diagrams should help more in unfamiliar domains. This aspect deserves future investigations. Researchers are particularly interested in such a future research.

- When trying to balance the cost related to the adoption of a new notation, one should also take into account the cost for training stakeholders. In this case, the cost of training is low because object diagrams are well-known. The practitioner is particularly interested in this outcome.

- The adoption of object diagrams as complement to class diagrams does not require a complete and radical process change in any company interested in doing so. This aspect is clearly relevant for the practitioner.

- Although the models were realistic for small-sized software projects, we are not sure that achieved results scale to real software projects. This point is relevant for the practitioner, but it is even more relevant for the researcher interested in assessing whether our results hold also in real projects.

6. Threats to Validity

Internal validity threats concern factors that may affect results in an undesirable way. To deal with this concern, we have properly chosen the experimental design in UniBas1, UniGe1 and PoliTo2 (within-participants counterbalanced design). To limit fatigue effect, we also introduced a break between the two tasks in UniBas1. Fatigue threat is not present in PoliTo2 since only two experimental objects were used by each participant (instead of four). Other possible threats concern information exchanged among participants within the same experiment and among experiments. This was prevented by monitoring participants while performing experimental tasks and by asking them back all material they used. Finally, to avoid apprehension, students were informed that they were not assessed on their performance.

External validity concerns the generalization of results. Although the models might appear small, their size has been chosen in accordance to software engineering book recommendations (e.g., [42]). In addition, we chose diagrams to be realistic for small/medium sized comprehension tasks. Therefore, we can argue that used UML models were simple, without being trivial. They were also related to four different domains (file systems, roads, trains, and catalogues). Future empirical investigations are needed to reduce external validity threats. To this end, we plan to conduct case studies on larger and more complex UML models and software. Different and special conceived empirical investigations (i.e., case studies) are needed to assess whether the obtained results hold also in real-sized projects. Our results pose the basis for such a kind of empirical studies, justifying their need. The use of students as participants may also affect external validity. However, they were trained on software engineering tasks and the execution of these tasks did not require a high level of industrial experience. Then, we can claim that the use of students here is appropriate as suggested in the literature [43, 44]. Working with students has also advantages, such as their prior knowledge being rather homogeneous, so to give a chance to test experimental and initial hypotheses [45] reducing failure risks and related costs. An additional advantage of using students is that the cognitive complexity of the objects under study is not hidden by participants experience. However, the participants in PoliTo1 and UniBas1 could be considered close to junior software developers. Moreover, Kitchenham et al. [46] argue that using students as participants instead of software engineers is not a major issue, as long as research questions are not specifically focused on experts.

Construct validity threats concern relationships between theory and observations. This kind of threat is related to how comprehension of UML models and time were measured. For comprehension, we used comprehension questionnaires. We carefully defined the questions so that they were neither too complicated, to make tasks impossible to perform, nor too simple, to make it difficult to observe any difference among participants. Answers were quantitatively evaluated using two different approaches: score in PoliTo1 and in PoliTo2 and F-measure in UniBas1 and in UniGe1. These approaches avoid as much as possible any subjective evaluation [11, 18, 47]. As far as IT experience and UML familiarity are concerned, used ordinal measures are a proxy of actual participants' experience and UML familiarity. Therefore, it is possible that a different measure could lead to different results. Comprehension effort was measured by means of time sheets and was validated qualitatively by researchers. We used this approach because very common in the empirical software engineering community.

Conclusion validity threats concern relationships between treatment and findings. In our family of experiments, we applied non-parametric tests to statistically reject null hypotheses. We opted for non-parametric tests because of sample size and non-normality of data. For rejecting null hypotheses, we exploited the Mann-Whitney and Fisher tests, because they are very robust and sensitive, even with small datasets [48]. In particular, the Fisher test is more accurate than the χ^2 test for small sample sizes. To compare the results of the four experiments we: (1) applied some transformations to discretize the comprehension level computed as F-measure (only for UniBas1 and UniGe1), (2) built a contingency matrix of the correct/wrong answers vs. the presence/absence of object diagrams to compare PoliTo2 with the other experiments, (3) for each experiment computed the odds ratio and 95% confidence interval using the contingency table. Given the nature of factors and the lack of other empirical evidence, we used the simplest possible model (i.e., the linear model) to study the effect of the IT Experience and UML familiarity context factors. Moreover, the post-experiment questionnaire (intended to get qualitative insights) was designed using standard ways and scales [35]. Finally, we ensure replicability of our experiments by providing an experimental package on the web.

7. Conclusions

In this paper, we report results from a family of controlled experiments aimed at investigating the combined effect of object and class diagrams on comprehensibility of software design. In each experiment, the participants were provided with either class diagrams alone or class and object diagrams together. Participants in our family of experiments were bachelor and master students having different IT experience and familiarity with UML.

Results suggested that participants having different IT experience and UML familiarity achieved different levels of benefit from the use of object diagrams. Our findings partially contrast the common beliefs: higher experienced participants having good UML familiarity benefited more of object diagrams in terms of comprehension than lower experienced participants. We speculate that this result depends on participants' capability to integrate different sources of knowledge, in our case class diagrams and objects diagrams, that is good for experienced participants and bad for inexperienced ones. As for comprehension effort, our results showed the absence of substantial variations in the effort needed to accomplish a comprehension task.

Based on our experimental results, a project manager should carefully think about the potential benefits of object diagrams considering her/his own industrial context. In a context where the software is developed/maintained by high experienced people with a good UML level, object diagrams could be used because useful. In this respect, obtained results justify and draw the need of future empirical studies to assess cost/benefit in the use of object diagrams to improve design comprehensibility. On the other hand, if the software is developed/maintained mainly by low experienced people with modest UML knowledge, it might not be convenient to adopt object diagrams.

The experiments belonging to our family of experiments involved participants possessing different UML familiarity and experience levels and made the use of UML models representing software systems belonging to different domains. Despite such a prominent variety, further empirical studies are highly desirable to provide further evidence or to contradict our results.

8. Acknowledgments

The authors would like to thank all the participants in the experiments of our family.

References

[1] OMG, Unified modeling language (OMG UML) specification, version 2.4.1, Tech. rep., Object Management Group (May 2011).

[2] B. Bruegge, A. H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, 3rd Edition, Prentice-Hall, 2010.

[3] M. Fowler, S. Kendall, UML Distilled: A Brief Guide to the Standard Object Modeling Language (4th ed.), Addison-Wesley Professional, 2010.

[4] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, R. Pretorius, Empirical evidence about the UML: a systematic literature review, Software: Practice and Experience 41 (4) (2011) 363–392.

[5] M. Torchiano, Empirical assessment of UML static object diagrams, in: Proceedings of the 12th IEEE International Workshop on Program Comprehension, IWPC 2004, IEEE Computer Society, Washington, DC, USA, 2004, pp. 226–230.

[6] G. Scanniello, F. Ricca, M. Torchiano, On the effectiveness of the UML object diagrams: A replicated experiment, in: Proceedings of the 15th International Conference on Evaluation and Assessment in Software Engineering, EASE 2011, IET Digital Library, 2011, pp. 76–85.

[7] S. N. Woodfield, H. E. Dunsmore, V. Y. Shen, The effect of modularization and comments on program comprehension, in: Proceedings of the 5th International Conference on Software Engineering, ICSE 1981, IEEE Press, Piscataway, NJ, USA, 1981, pp. 215–223.

[8] R. Agarwal, P. De, A. Sinha, Comprehending object and process models: an empirical study, IEEE Transactions on Software Engineering 25 (4) (1999) 541–556.

[9] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, G. Tortora, Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments, IEEE Trans. on Soft. Eng. 39 (3) (2013) 327–342.

[10] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, C. Britton, UML class diagram syntax: an empirical study of comprehension, in: Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation - Volume 9, APVis 2001, Australian Computer Society, Inc., Darlinghurst, Australia, 2001, pp. 113–120.

[11] M. Staron, L. Kuzniarz, C. Wohlin, Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments., Journal of Systems and Software 79 (5) (2006) 727–742.

[12] G. Scanniello, C. Gravino, M. Genero, J. A. Cruz-Lemus, G. Tortora, On the impact of UML analysis models on source-code comprehensibility and modifiability, ACM Trans. Softw. Eng. Methodol. 23 (2) (2014) 13:1–13:26.

[13] M. Leotta, F. Ricca, G. Antoniol, V. Garousi, J. Zhi, G. Ruhe, A pilot experiment to quantify the effect of documentation accuracy on maintenance tasks, in: Proceedings of the 29th International Conference on Software Maintenance, ICSM 2013, IEEE, 2013, pp. 428–431. `doi:10.1109/ICSM.2013.64`.
URL `http://dx.doi.org/10.1109/ICSM.2013.64`

[14] S. Yusuf, K. Kagdi, J. I. Maletic, Assessing the comprehension of UML class diagrams via eye tracking, in: Proceedings of the 15th International Conference on Program Comprehension, ICPC 2007, IEEE Computer Society, Washington, DC, USA, 2007, pp. 113–122.

[15] A. De Lucia, C. Gravino, R. Oliveto, G. Tortora, An experimental comparison of ER and UML class diagrams for data modelling, Empirical Software Engineering 15 (5) (2010) 455–492.

[16] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, UML collaboration diagram syntax: An empirical study of comprehension, in: Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis, VISSOFT 2002, IEEE Computer Society, Washington, DC, USA, 2002, pp. 13–22.

[17] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, M. Ceccato, The role of experience and ability in comprehension tasks supported by UML stereotypes, in: Proceedings of the 29th International Conference on

Software Engineering, ICSE 2007, IEEE Computer Society, Minneapolis, MN, USA, 2007, pp. 375–384.

[18] F. Ricca, M. Di 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 Transactions on Software Engineering 36 (1) (2010) 96–118.

[19] A. Gemino, D. Parker, Use case diagrams in support of use case modeling: Deriving understanding from the picture, Journal of Database Management 20 (1) (2009) 1–24.

[20] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca, A. Susi, Comparing the comprehensibility of requirements models expressed in use case and Tropos: Results from a family of experiments, Information and Software Technology 55 (10) (2013) 1823–1843.

[21] S. Xie, E. Kraemer, R. E. K. Stirewalt, Empirical evaluation of a UML sequence diagram with adornments to support understanding of thread interactions, in: Proceedings of the 15th IEEE International Conference on Program Comprehension, ICPC 2007, IEEE Computer Society, Washington, DC, USA, 2007, pp. 123–134.

[22] J. A. Cruz-Lemus, M. Genero, D. Caivano, S. Abrahao, E. Insfran, J. A. Carsi, Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments, Information and Software Technology 53 (12) (2011) 1391–1403.

[23] C. Gravino, G. Scanniello, G. Tortora, An empirical investigation on dynamic modeling in requirements engineering, in: Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems, MoDELS 2008, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 615–629.

[24] C. Glezer, M. Last, E. Nachmany, P. Shoval, Quality and comprehension of UML interaction diagrams - an experimental comparison, Information and Software Technology 47 (10) (2005) 675–692.

[25] J. Cruz-Lemus, M. Genero, M. Manso, S. Morasca, M. Piattini, Assessing the understandability of UML statechart diagrams with composite

states – a family of empirical studies, Empirical Software Engineering 14 (6) (2009) 685–719.

[26] A. Gross, J. Doerr, EPC vs. UML activity diagram - two experiments examining their usefulness for requirements engineering, in: Proceedings of the 17th IEEE International Conference on Requirements Engineering, RE 2009, IEEE, Los Alamitos, CA, USA, 2009, pp. 47–56.

[27] G. Reggio, F. Ricca, G. Scanniello, F. D. Cerbo, G. Dodero, On the comprehension of workflows modeled with a precise style: results from a family of controlled experiments, Software & Systems Modeling 14 (4) (2013) 1481–1504.

[28] G. Reggio, M. Leotta, F. Ricca, E. Astesiano, Business process modelling: Five styles and a method to choose the most suitable one, in: Proceedings of the 2nd International Workshop on Experiences and Empirical Studies in Software Modelling, EESSMod 2012, ACM, 2012, pp. 8:1–8:6. `doi:10.1145/2424563.2424574`.
URL `http://dx.doi.org/10.1145/2424563.2424574`

[29] L. Thomas, M. Ratcliffe, B. Thomasson, Scaffolding with object diagrams in first year programming classes: some unexpected results, in: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, ACM, New York, NY, USA, 2004, pp. 250–254.

[30] N. Juristo, A. Moreno, Basics of Software Engineering Experimentation, Kluwer Academic Publishers, 2001.

[31] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Kluwer, 2012.

[32] V. Basili, G. Caldiera, D. H. Rombach, The Goal Question Metric Paradigm, Encyclopedia of Software Engineering, John Wiley and Sons, 1994.

[33] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object Oriented Software, Addison-Wesley, 1995.

[34] J. Cohen, Statistical power analysis for the behavioral sciences (2nd ed.), Lawrence Earlbaum Associates, Hillsdale, NJ, 1988.

[35] A. N. Oppenheim, Questionnaire Design, Interviewing and Attitude Measurement, Pinter, London, 1992.

[36] E. Kamsties, A. von Knethen, R. Reussner, A controlled experiment to evaluate how styles affect the understandability of requirements specifications., Information and Software Technology 45 (14) (2003) 955–965.

[37] G. Scanniello, M. Staron, H. Burden, R. Heldal, On the effect of using SysML requirement diagrams to comprehend requirements: results from two controlled experiments, in: International Conference on Evaluation and Assessment in Software Engineering, EASE 2014, ACM, 2014, pp. 49:1–49:10.

[38] A. Agresti, An Introduction to Categorical Data Analysis, Wiley-Interscience, 2007.

[39] V. Basili, F. Shull, F. Lanubile, Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25 (4) (1999) 456–473.

[40] J. C. Carver, N. J. Juzgado, M. T. Baldassarre, S. Vegas, Replications of software engineering experiments, Empirical Software Engineering 19 (2) (2014) 267–276.

[41] B. Kitchenham, H. Al-Khilidar, M. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, L. Zhu, Evaluating guidelines for reporting empirical software engineering studies, Empirical Software Engineering 13 (2008) 97–121.

[42] B. Bruegge, A. H. Dutoit, Object-Oriented Software Engineering: Conquering complex and changing systems, Prentice-Hall, 2000.

[43] J. Carver, L. Jaccheri, S. Morasca, F. Shull, Issues in using students in empirical studies in software engineering education, in: Proceedings of the 9th International Software Metrics Symposium, METRICS 2003, IEEE Computer Society, Washington, DC, USA, 2003, pp. 239–249.

[44] M. Svahnberg, A. Aurum, C. Wohlin, Using students as subjects - an empirical evaluation, in: Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2008, ACM, New York, NY, USA, 2008, pp. 288–290.

[45] D. I. K. Sjoberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N. Liborg, A. C. Rekdal, A survey of controlled experiments in software engineering, IEEE Transactions on Software Engineering 31 (9) (2005) 733–753.

[46] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721–734.

[47] L. C. Briand, Y. Labiche, M. Di Penta, H. Yan-Bondoc, An experimental investigation of formality in UML-based development, IEEE Transactions on Software Engineering 31 (10) (2005) 833–849.

[48] H. Motulsky, Intuitive biostatistics: a Nonmathematical guide to statistical thinking, Oxford University Press, 2010.