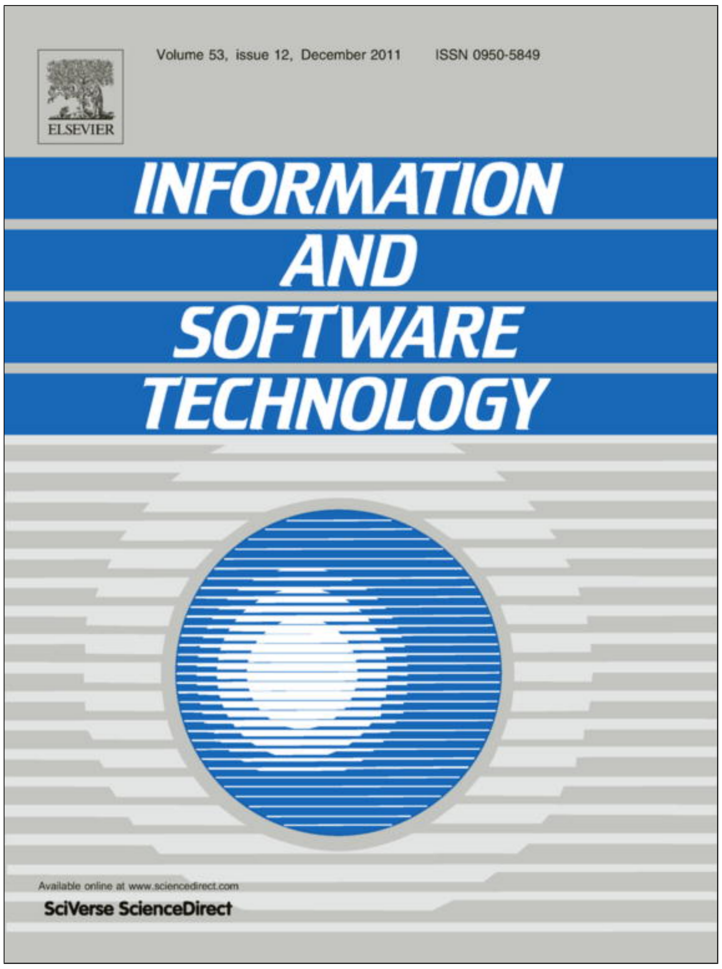

Volume 53, issue 12, December 2011
ISSN 0950-5849
ELSEVIER
INFORMATION AND SOFTWARE TECHNOLOGY
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Information and Software Technology 53 (2011) 1391–1403

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments

José A. Cruz-Lemus [a,*], Marcela Genero [a], Danilo Caivano [b], Silvia Abrahão [c], Emilio Insfrán [c], José A. Carsí [c]

[a] ALARCOS Research Group, Department of Technologies and Information Systems, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain
[b] Department of Informatics, University of Bari, Via E. Orabona, 4, 70126 Bari, Italy
[c] ISSI Research Group, Department of Information Systems and Computation, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

A R T I C L E   I N F O

Article history:
Received 7 June 2010
Received in revised form 27 June 2011
Accepted 21 July 2011
Available online 3 August 2011

Keywords:
Stereotypes
UML sequence diagrams
Comprehension
Family of experiments
Meta-analysis

A B S T R A C T

Context: The conventional wisdom states that stereotypes are used to clarify or extend the meaning of model elements and consequently should be helpful in comprehending the diagram semantics.
Objective: The main goal of this work is to present a family of experiments that we have carried out to investigate whether the use of stereotypes improves the comprehension of UML sequence diagrams.
Method: The family of experiments consists of an experiment and two replications carried out with 78, 29 and 36 undergraduate Computer Science students, respectively. The comprehension of UML sequence diagrams with and without stereotypes was analyzed from three different perspectives borrowed from the Cognitive Theory of Multimedia Learning (CTML): semantic comprehension, retention and transfer. In addition, we carried out a meta-analysis study to integrate the different data samples.
Results: The statistical analysis and meta-analysis of the data obtained from each experiment separately indicates that the use of the proposed stereotypes helps improving the comprehension of the diagrams, especially when the subjects are not familiar with the domain.
Conclusions: The set of stereotypes presented in this work seem to be helpful for a better comprehension of UML sequence diagrams, especially with not well-known domains. Although further research is necessary for strengthening these results, introducing these stereotypes both in academia and industry could be an interesting practice for checking the validity of the results.

1. Introduction

The Unified Modeling Language (UML) [1] is a widely-accepted standard notation for expressing models within the software development process. However, the main drawback of this language is that, due to its general-purpose nature, the set of general modeling constructs it provides may not be suitable for a specific purpose and hence the modeling might not be effective [2]. To address this drawback, several studies propose the improvement of modeling with UML by customizing it with the extension mechanisms inherent in the language (e.g., stereotypes, OCL constraints, and tag values). In particular, stereotypes are used to clarify or extend the meaning of model elements and to introduce new modeling elements, reusing the syntax of similar elements already available in the language.

In our experience, stereotypes are often used in industrial contexts and their use spans from use cases to class diagrams. Indeed, companies use stereotypes within their development processes to specialize general processes aiming to fit them to a particular technology in use, such as programming languages (e.g., C#, Java), application type (e.g., real time, Web applications, client–server, standalone), reusable components used (e.g., Microsoft Foundation Class Library, Enterprise Java Beans Library) or simply to give more detailed guidelines to the practitioners involved in the system development processes. The use of stereotypes is also widespread in academia, giving students some practical instruments for guiding software design and filling the information and representation gap of UML standard language. Well-known stereotypes widely used in industry and academic contexts are Conallen's stereotypes [3], which are also supported by the most extensively-used UML Case Tools, such as ARGO UML, Enterprise Architect and STAR UML. Today, these case tools are widely employed in industrial contexts.

Although the use of stereotypes has already been widely accepted in software modeling, it is interesting to have empirical evidence about whether they really contribute to building better models. In recent years, several studies addressing this question have been conducted in the field of Software Engineering [2,4]. They focus on investigating the influence of stereotypes on the

* Corresponding author. Tel.: +34 926295300 6217; fax: +34 926295354.
E-mail addresses: JoseAntonio.Cruz@uclm.es (J.A. Cruz-Lemus), Marcela.Genero@uclm.es (M. Genero), caivano@di.uniba.it (D. Caivano), sabrahao@dsic.upv.es (S. Abrahão), einsfran@dsic.upv.es (E. Insfrán), pcarsi@dsic.upv.es (J.A. Carsí).

0950-5849/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2011.07.002

comprehension of UML class and collaboration diagrams. The reason for focusing on comprehension is because it is widely recognized that comprehension is one of the main factors influencing maintainability [5–8], and it must be assessed from the first steps of the software lifecycle, in the modeling phase (see related work in Section 2). An UML diagram must be well understood before any change can be introduced when maintaining it.

Nevertheless, the influence of stereotypes on the comprehension of requirement models, such as UML sequence diagrams, has not been investigated yet. This fact has motivated us to develop the research presented in this work. We focus on UML sequence diagrams, since they are a widely used technique for reasoning about object interactions needed to realize a given scenario of a functional requirement. In addition, we consider the comprehension of these diagrams, since this is essential in the validation of requirement specifications among developers and stakeholders. A sequence diagram must first be understood before any desired changes to it can be identified, designed, or implemented.

The main goal of this paper, therefore, is to present a family of experiments to investigate whether the use of stereotypes improves the comprehension of UML sequence diagrams. Specifically, we consider a set of stereotypes that have been proposed to enrich the semantics of interaction messages in UML sequence diagrams in the context of a Requirements Engineering approach for model-driven software development [9,10]. The benefit of using these stereotypes is twofold. Firstly, they can improve the comprehension of UML sequence diagrams, and secondly, they provide specific information about how to transform each individual source element in the transformation of UML sequence diagrams into conceptual models in a model-driven development process.

We started investigating our hypotheses by means of a controlled experiment, as in [11]. Our decision to address the above issues by using controlled experiments stems from the many confounding and uncontrollable factors that could blur the results in an industrial context. In addition, during the earlier stages of an investigation, controlled experiments enable the investigators to understand the issues at stake better, as well as the factors to be considered. Furthermore, controlled experiments enable the assessment of whether the results obtained on smaller artifacts and tasks can at least be considered encouraging and if they can justify further evaluation in more realistic settings.

A total of three empirical studies were conducted. The original experiment was carried out with 78 Computer Science undergraduate students at the University of Bari in Italy. Two replications of this experiment were then conducted at the University of Castilla-La Mancha in Spain to confirm the original findings, with 29 and 36 Computer Science undergraduate students. The goal was to provide evidence for the generalization of the results by repeating the experiment in different environments (using different subjects or materials).

As well as the family of experiments, we performed a meta-analysis to aggregate the results obtained in the individual studies. Meta-analysis has been recognized as an appropriate way to aggregate or integrate the findings of empirical studies in order to build a solid body of knowledge on a topic based on empirical evidence [12,13]. Moreover, the need for meta-analysis is gaining relevance in empirical research, as is demonstrated by the fact that it is a recurrent topic in various forums related to Empirical Software Engineering. In other areas, such as psychology or medicine, a single study is extremely unlikely to be definitive. Dozens and even hundreds of studies on the same topic may follow. In Empirical Software Engineering, it is unusual for a large amount of studies concerning the same topic to take place, but it is necessary to cross the borders of individual studies to extract conclusions of a more general kind from families of experiments, with or without significant results. This is the reason why we also performed a meta-analysis of our data.

This paper is organized as follows. Firstly, related works on the empirical evaluation of UML diagram comprehension and specifically with regard to the use of stereotypes are presented in Section 2. The definition of stereotypes for UML sequence diagrams is explained in Section 3. The description, execution and data analysis of the family of experiments is described in Section 4. A meta-analysis to provide a global analysis of the individual experiments is presented in Section 5. Finally, the conclusions and a discussion on future research work are set out in Section 6.

2. Related work

The most empirically-studied quality attribute is undoubtedly comprehensibility, especially in UML diagrams [5,6,14–20]. The explanation for this is that the ease with which respective UML diagrams can be understood affects how they must be maintained, tested, etc.

To be more specific, with respect to the use of stereotypes in the comprehension of UML diagrams, which is the focus of this piece of work, there are several relevant publications in the literature. The following should be highlighted, among others:

- Staron et al. [21] reported a controlled experiment aimed at evaluating whether stereotypes improve efficiency and effectiveness of reading techniques for software inspections. The results showed that the presence of stereotypes improves both aspects of the reading techniques. The efficiency is increased by 76% and the effectiveness by 17%. It was their intention to verify whether the increased comprehension of software artefacts results in improvements of inspection techniques. The results from this experiment seem to support the hypothesis that it did indeed do so. In other words, the experiment provides evidence that the presence of stereotypes in UML designs improves the correctness of models after the inspection process, since stereotyped models contained fewer faults (more faults were found in the models during inspections). An additional outcome of the study is the identification of a reading technique which is potentially the most suitable one for inspecting UML designs with stereotypes. As it appears, the Checklist-Based Reading technique is the most efficient technique and the Perspective-Based Reading technique is the most effective one.
- Staron et al. [2] presented a set of four controlled experiments conducted both in academia and in industry, on a total of 72 subjects. The results of all four experiments, aimed at evaluating the role of stereotypes in understanding UML models, confirm that the use of stereotypes improves UML model comprehension and show the magnitude of the improvements. Improvements were achieved in the following three aspects: an increase in the number of correct answers in the tests checking the level of understanding, a decrease in the time required for answering the questionnaire, and a decrease in the relative time for a correct answer. The first experiment was conducted in academia with the largest number of subjects. It was also replicated with industrial subjects in the fourth experiment, to obtain industrial validity of the results. The order of presenting stereotyped and non-stereotyped models was chosen in such a way as to be different for each experimental group. Since this order of the presentation of the models in experiment rounds could influence the results, another experiment was conducted. This changed the order of model presentation.. The results showed that the order could influence the results, but that there was still improvement after introduction of stereotypes. Furthermore, the graphical icons used for representation of stereotypes in experiment objects could be the main factor in the improvement. To address this threat, the researchers conducted

the third experiment, in which the icons were replaced by a text. Even in this case, the introduction of stereotypes showed improvements. Finally, the fact that the study subjects were students could be one of the factors influencing the results, so the fourth experiment was conducted in the same way as the first experiment, with subjects that were professionals in industry. Considerable improvements were achieved in all four experiments for stereotypes that were represented both as text and with icons. The stereotypes with their graphical representation as icons improve the understanding of UML models more than stereotypes represented as textual adornments of model elements. The claim is based on the evaluation of the hypothesis with statistical significance testing for three experiments and a qualitative analysis for the industrial experiment. The largest improvement was achieved in the case of the industrial experiment. The results of the industrial experiment were in accordance with the expectations based on the division into different types of subjects. Thus, although the number of professionals participating in this study was small, the results were fully aligned with those of the original study and therefore it was regarded as valid.
- Ricca et al. [4] reported results from a family of experiments aimed at investigating the effect of stereotypes in software comprehension. In particular, the experimentation consisted of three replications, involving both graduate and undergraduate subjects, in which the task was the understanding of a Web application documented with Conallen's stereotyped class diagrams or pure UML class diagrams (control group). The level of ability of the subjects involved was ranked according to the scores they had obtained in previous courses. Results indicate that the use of stereotypes is not the only factor which influences the comprehension performance. Non significant results are obtained when this factor is considered alone. On the other hand, combining this factor with the degree of experience and ability, results become (statistically) significant and much more interpretable. Interesting implications can be derived for the practitioners. Novice users obtain most benefits from stereotyped design diagrams and their performance shows smaller variability than when using standard UML only. Experienced users are likely to prefer more traditional and standard information sources. Derivation of the missing information is probably not a major impediment for them. Results suggest that organizations employing developers with low experience can achieve a significant improvement in performance by adopting stereotyped UML diagrams for Web applications.

The literature review described above reveals that comprehension of stereotyped diagrams is a principal concern in the context of UML modeling, but that the contribution of stereotypes with which to improve the comprehension of UML sequence diagrams has not been investigated yet. This fact has motivated us to carry out the family of experiments presented in the current paper. The first experiment of the family has been presented previously [22], but it will not be referred to as related work, since it will be introduced in Section 4 together with the rest of studies of the family to make the comprehension of the whole family easier.

Research in all the literature reviewed has influenced the design of this study, as we used the experiences reported and best practices to design the experiment.

3. Stereotypes and UML sequence diagrams

UML indicates that sequence diagrams are a means to model an aspect of the dynamic behavior of a system [23]. They can be used in the context of the whole system, or of a subsystem; they can be attached to a Use Case, or to an object service. Some authors indicate that at least one sequence diagram should be drawn per Use Case [23,24], in order to describe the expected behavior of the Use Case. When a sequence diagram is developed for a Use Case, the Use Case description can be employed to develop at least the initial draft of the sequence diagram. Throughout the design process the Use Case diagram can be revised based on the results of the sequence diagram, and vice versa, until both models are appropriately tuned [25].

A sequence diagram has two dimensions: the vertical dimension represents time, and the horizontal dimension represents the different object types (classes). Time proceeds down the page and there is no significance to the horizontal ordering of the object types.

Lamsweerde [26] draws the distinction between specification of interactions at type level or at instance level. When using a sequence diagram at the *type level*, it represents a pattern interaction, which is a set of messages among object types with which to carry out behavior. When using a sequence diagram at *instance level*, it represents a concrete set of object interactions, i.e., an example of a type level scenario.

The emphasis of type level sequence diagrams is on graphically representing the pattern interaction between object types by sending and receiving messages as time advances. In addition, when using type level sequence diagrams to realize a Use Case, the focus should not be on specifying detailed behavior with complex iterations and conditional messages (the *how*) but on the interactions that are needed to perform the Use Case purpose (the *what*). This difference means that when specifying the Use Case behavior at the type level not all the interactions are needed to be represented in the pattern interaction but the essential ones (the specification of those interactions representing design decisions and control flow are deferred to the design phase).

There must be one sequence diagram per Use Case. In the case of alternative course of actions (if any), they will be represented as fragments of the sequence diagram using the operator fragment "alt" with the corresponding guard expression, following the UML 2.2 notation. To build the corresponding sequence diagram for a Use Case at the type level, we analyze each Use Case at two levels:

- Use Case diagram level: actors that communicate with the Use Case.
- Use Case specification level: the set of steps or responsibilities to be performed to accomplish the Use Case.

At the Use Case diagram level, an actor can send and/or receive information to/from the Use Case. Every Use Case has at least one actor, which is the Use Case initiator, along with other collaborating actors. All the actors are potentially object types to be taken into account for the sequence diagram.

At the Use Case specification level, the main task is to obtain the set of responsibilities implied by the corresponding Use Case steps in the Use Case description. Essentially, when the Use Case receives a stimulus (an external interaction), the system produces a set of interactions between its internal components as a response. These interactions are represented as messages among types of objects (i.e., classes) in the sequence diagram.

As defined in UML 2.2, "the profile mechanism has been specifically defined for providing a lightweight extension mechanism to the UML standard" [1]. A part of a profile definition is a set of stereotypes. A stereotype is a new modeling element defined by extending one existing element in the UML meta-model. Stereotypes provide more information in a given context of use than that given by the usual modeling element.

In [9], a specific classification for sequence diagram messages is presented. This classification uses stereotypes to distinguish the type of the messages used in the sequence diagrams according to the effect of the message. The main purpose of providing these stereotypes is to improve the comprehension of sequence diagrams according to the nature of the interactions, in addition to the syntactical names that can be assigned to them. According to a classification of stereotypes in object-oriented languages [27], these stereotypes can be considered as a *restrictive* stereotype since they do not change the semantics of the base element (message) but extend and modify it.

The four proposed stereotypes are:

- "signal". It is used with messages that represent interactions between actors and the system interface. Actors communicate with the system, sending stimuli to the system interface. The system interface sends back answers to the actors.

- "service". It is used with messages that represent the change of the internal state of an object of the receiving object type. In this type of messages, the source of the message can be an object type, or the system interface. The target of the message is always an object type. Three different kinds of services have been identified: the *creation* of an object, the *destruction* of an object, and the *update* of an object state. In order to identify the kind of service, three tags can be attached to the stereotype: {new}, {destroy}, and {update}, respectively.
- "query". It is used with messages that represent queries about other objects or about class population. In the context of a sequence diagram, due to the encapsulation of the object, this kind of interaction is the only way of knowing the state of other objects. It is also important to express the multiplicity of the expected answer by using a tag value (e.g., 0...1, 1...1).
- "connect". It is used with messages that capture an important kind of object interaction (although sometimes it is difficult to identify this kind of interaction at first sight). In the object-oriented model, there are many ways that an object can be related to other objects (e.g., associations, aggregations, compositions). In particular, when dealing with object interactions in early phases of the software development, we need to recognize if an object can be a part of another object or if it can just be associated to (related to) other objects. Due to the encapsulation property, these are the only way an object can have access to another object's internal. For this reason, when an object is created (or later in the object lifecycle) we must know which other objects are related to it. This kind of interaction is specified using the "connect" stereotype. It is also important to express the multiplicity of this interaction by using a tag value (e.g., 0...1, 1...1, 1...∗) meaning how many objects are needed (optional one, exact one, one or many, etc.).

Although the UML 2.2 standard [1] defines a set of general predefined stereotypes that can be used in UML (annex. C of the UML Superstructure – Standard Stereotypes), depending on the L2 or L3 level of compliance with the UML, there are no specific stereotypes for sequence diagrams. For this reason, specific methods or tools should extend the notation and meaning provided by the standard UML features if they need to have a semantically richer modeling notation.

In Fig. 1, we present an example which shows the stereotyped interactions (messages) that are needed to realize a Use Case at the type level for selling products (items) in a conventional store. The explanation of the main interaction/messages is also given below. Please note that, due to the emphasis on specifying what is needed to realize the Use Case (the *what* not the *how*), control flow interactions are not specified.

- *Messages 1 and 2.* The cashier starts a sale. The first message represents the beginning of communication between the actor *Cashier* and the *Interface* of the software system. The following message *introduce_sale_data* provides the data which is necessary to perform the operation. The stereotype used in both messages is "signal".
- *Message 3.* A new sale is created. This message implies the responsibility of creating a new sale and registering its information. We have named this responsibility *create_sale* and allocated it to a class called *Sale*. As the purpose of this message is the creation of a new object, the stereotype used is "service" with the tag value {new}.
- The loop block marks a group of messages that are repeated at least one time. At this stage of the development process it is not important to specify at the maximum level all the implementation details. But, if it would be necessary the analyst could introduce some explanation in natural language when specifies the minimum and the maximum multiplicity.
- *Message 4.* Each item to be sold has to be recorded. This action implies the responsibility of registering information about the items to be sold, the quantity, the discount, etc. We have named this responsibility *create_sale_line* and allocated it to a class called *Sale-line*. As a new object has to be created in order to store some information, the stereotype used is "service" with the tag value {new}.
- *Message 5.* The message *select_one_item* (with the stereotype "connect" and the tag value {1...1}) is identified because the analyst determines that a *Sale_Line* must be related to one and only one *Item* (product) assigned to it.
- *Message 6.* The system determines the item price and adds this information to the current *Sale_Line*. This action implies the responsibility of querying the object that stores the item price. We have named this responsibility *get_item_info* with the stereotype "query" and the tag value {1...1}.
- *Message 7.* Another responsibility of the Use Case is that of updating the item stock when the item is sold. We have named this responsibility *change_stock* and allocated it to the previously identified class *Item*. As the purpose of this service is to change some data about one object, the stereotype used is "service" with the tag value {update}.
- *Message 8.* At this point of the interaction, the actor *Cashier* ends the sell by registering the items (products) to sell.
- *Message 9.* The system calculates the total amount of the sale from the corresponding set of *Sale_Lines*. This action implies the responsibility of querying the set of *Sale_Lines*. We have named this responsibility *get_subtotals* with the stereotype "query" and the tag value {1...∗}. The result is stored in a variable named *total*.
- *Message 10.* The system then shows the information concerning the total of the sale to the *Cashier*.

Note that on specifying a sequence diagram at the type level, details such as parameters specification, iteration or control flow are not the main focus, since we are mainly identifying and specifying what interactions are needed to realize the Use Case. However, this detailed specification is considered optional in the sense that depending on the context or on the point of view of the analyst it can help to improve the understanding of the Use Case. The data that the messages convey is very important but the sequence diagrams do not focus on the manipulation of data even though data can be used to decorate the diagrams. This is in line with the UML Superstructure specification [1] that recognizes that sometimes traces in the interaction specification do not tell the complete story due to legal or other reasons. In our case, the reason is that the interaction specification of sequence diagrams has the main goal of identifying the relevant participants

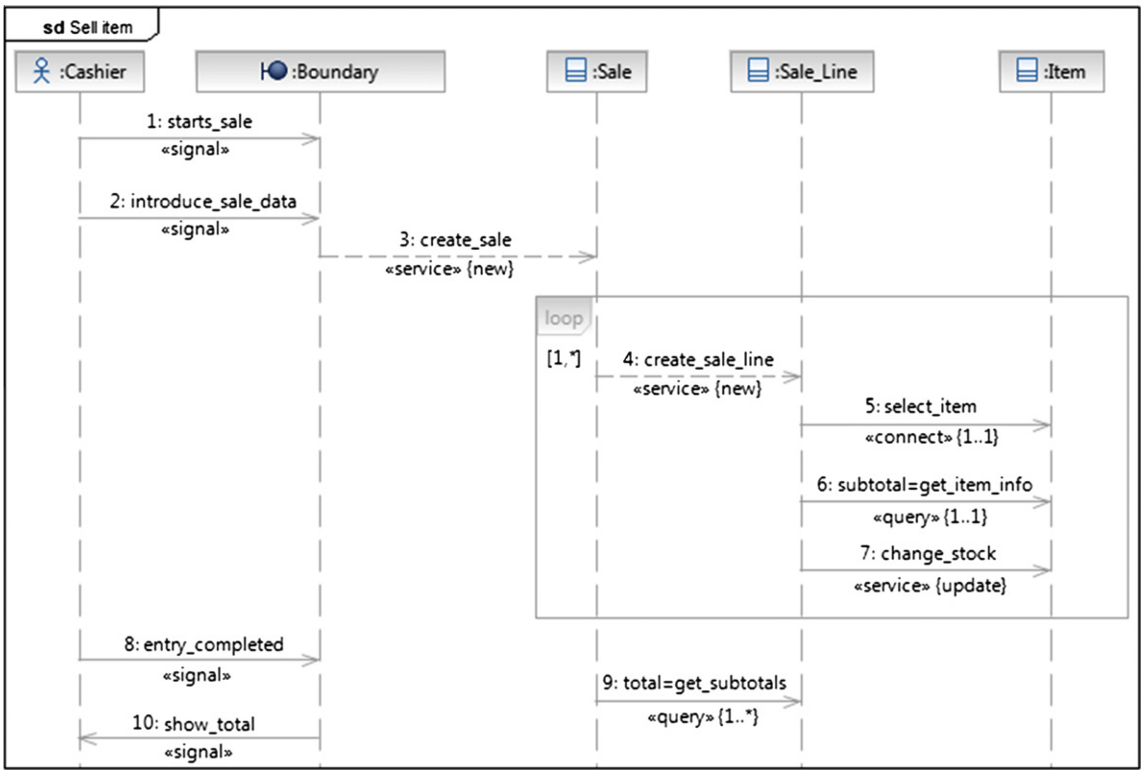


Fig. 1. An example of a sequence diagram with stereotyped messages.

understanding their essential interactions. This activity leads us to discover classes and services and not to describe the complete behavior of the object types involved, which is responsibility of the designers. In addition, we also highlight that the specific style for describing and naming object types and interactions (messages) will always rely on the preferences, point of view, and understanding of the developers.

Finally, the process of specifying a Use Case as an interaction of collaborative objects continues thus until all the steps of the Use Case description are defined in its corresponding sequence diagram. Developers should verify this specification, looking for the satisfaction of the Use Case purpose and they ought also to validate it, together with users, during and after its construction. Comprehension is, therefore, a critical factor in the building and maintaining of this type of diagrams.

4. The family of experiments

This section explains the main characteristics of the family of experiments that has been carried out. The family consisted of a controlled experiment (EXP) [22] and two strict replications of it (REP1 and REP2), in which none of the dependent or independent variables vary [28], as shown in Fig. 2.

The main goal of this family of experiments is to investigate whether the use of stereotypes improves the understanding of UML sequence diagrams. Therefore, by using the GQM template for goal definition [28,29], the goal of our experiment is defined as follows: *"Analyze the use of stereotypes for the purpose of evaluating it with respect to the comprehension of UML sequence diagrams from the point of view of the researcher, in the context of undergraduate students in Computer Science from a couple of universities in Italy and Spain"*.

In order to run and report all the studies of this family of experiments, we followed the recommendations provided in several works [30,31]. The design of the experiments presented in this paper is similar to that presented in [2].

We selected a balanced factorial design in which the group-interaction acted as a confounding factor [32]. The objects were UML sequence diagrams, with two possible values – stereotyped diagram and non-stereotyped diagram.

All the subjects were randomly assigned to four groups (1–4). The experiments consisted of two rounds. Two different diagrams were

EXP	REP1	REP2
# students: 78 (Computer Science – 4th year) *language:* Italian *location:* University of Bari (Italy) *date:* February 2008	*# students:* 29 (Computer Science – 3rd year) *language:* Italian *location:* University of Castilla-La Mancha (Spain) *date:* April 2008	*# students:* 36 (Computer Science – 5th year) *language:* Spanish *location:* University of Castilla-La Mancha (Spain) *date:* April 2008

Fig. 2. Family of experiments road-map. The first replication (REP1) was performed by a set of Italian students that were attending a course at the University of Castilla-La Mancha (Spain), which is why the materials were presented to them in Italian.

presented to every subject in each group. To avoid a possible learning effect, the diagrams came from different application domains (A – a hotel room booking and B – extras rental in a car-rental).

4.1. The planning of the experiments

The main features of the planning are described next:

- *Subjects*. The number of subjects of each study was shown in Fig. 2.

The subjects had participated in two Software Engineering courses in which they had acquired training in UML diagrams. Their knowledge was sufficient for them to understand the non-stereotyped diagrams given, and they had roughly the same background. They had knowledge about the use of stereotypes in general, but they were taught about the stereotypes we proposed for UML sequence diagrams in a training session organized to take place the day before the experiment was carried out.

To avoid social threats due to evaluation apprehension, the students were not graded on their performance. The participants were, however, granted extra points in their final evaluation at the end of the course.

- *Experiment objects*. The experimental objects consisted of four diagrams, which is summarized below:
  - *DA-S*: stereotyped diagram A and a general description of each type of stereotype.
  - *DA-N*: non-stereotyped diagram A.
  - *DB-S*: stereotyped diagram B and a general description of each type of stereotype.
  - *DB-N*: non-stereotyped diagram B.

UML sequence diagram *A–x* described a car rental domain. It described an "Extras Rental" Use Case which may occur alongside a car rental (e.g., child seats, DVD player, Navigator). Diagram B-x described a hotel domain. It described a "Book room" Use Case which occurs when a guest rents a room in a hotel. These experimental objects were presented in Italian or Spanish, in order to avoid a possible negative language effect.

As an example, Appendix A presents the DA-N materials translated into English for the reader's convenience. The original and complete materials were presented in Italian and Spanish and can be found at http://alarcos.esi.uclm.es/ExpStereotypes.

- *Independent and dependent variables*. There are two independent variables in the family of experiments, the diagram type, with values: S (stereotyped) and N (non-stereotyped), and the diagram domain (A and B). By combining each level of the independent variables we obtain four treatments, as reflected by the four sequence diagrams which are the objects of the experiment.

Following certain suggestions concerning the measurement of comprehension [33–35], we have used the CTML [36]. This choice was made for several reasons. First of all, it focuses on words and graphics, which are the elements in the UML sequence diagrams grammar. Secondly, it provides principles for the design of effective multimedia presentations which can be tested empirically. Thirdly, it has evolved through years of work and development of experimental instruments and methods related to model comprehension [36,37].

By following the CTML, the comprehension of the UML sequence diagrams has been defined through three variables:

- *Semantic comprehension*: The ability to comprehend the semantics of the models.
- *Retention*: The comprehension of material being presented, and the ability to retain knowledge from it.
- *Transfer*: The ability to use the knowledge gained from the material to solve related problems which are not directly answerable from it.

It was expected that due to the use of stereotypes which enrich the meaning of the messages in the sequence diagram specification, the reader of a stereotyped sequence diagram will have key knowledge that is additional to what a reader of a non-stereotyped one has for comprehension, retention and transfer tasks.

To add clarity, the quality model used (defined according to the GQM approach [29,38]) is presented in Fig. 3, which also points out the experimental hypothesis defined for each measure used.

In order to measure these variables, we have used three separate tests based on questionnaires. Each comprehension measure was computed as:

- *Effectiveness*: The proportion of correct answers provided in each test (number of correct answers/number of questions). This measure reflects the ability to understand the material presented correctly.
- *Efficiency*: The proportion of correct answers divided by time (Effectiveness $*$ 100/Time).

In this work, we call these values SCEffec/SCEffic (Semantic Comprehension Effectiveness and Efficiency), TransEffec/TransEffic (Transfer Effectiveness and Efficiency), and finally, RetenEffec/RetenEffic (Retention Effectiveness and Efficiency).

- *Hypotheses*. For Semantic Comprehension we wished to test the following hypotheses:

Null hypotheses:

- $H_{1.1,0}$: stereotypes do not improve the subjects' *SCEffec* when attempting to comprehend a UML sequence diagram.
- $H_{1.2,0}$: stereotypes do not improve the subjects' *SCEffic* when attempting to comprehend an UML sequence diagram.

Alternative hypotheses:

- $H_{1.1,1} = \neg H_{1.1,0}$.
- $H_{1.2,1} = \neg H_{1.2,0}$.

We analogously formulated a set of hypothesis H2 for Retention measures (RetenEffec and RetenEffic), and another set H3 related to the Transfer measures (TransfEffec and TransEffic) (see Fig. 3).

- *Instrumentation*. The instruments used in the experiment were three tests attached to each of the four treatments. In each test the subjects were asked to write down the time before starting to solve the tasks required in each test.

The tests are described as follows:

- *Test 1*: contained 10 questions concerning the semantics of the diagrams. These questions can be answered using YES, NO (or left blank), and were formulated in similar way than in [6,34]. This task was used to obtain *SCEffec* and *SCEffic*.
- *Test 2*: consisted of a 'fill-in-the-blanks' task in which the subjects had to complete a test describing the functionality of the diagrams. This task was used to calculate *RetenEffec* and *RetenEffic*.

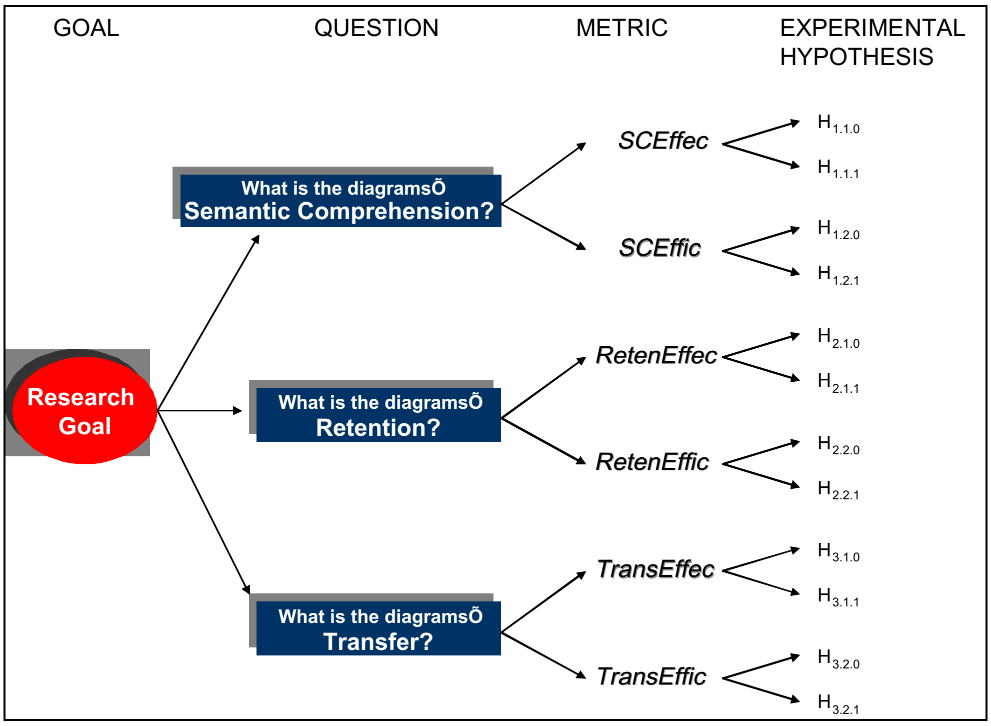


Fig. 3. Quality model used and experimental hypotheses.

– *Test 3*: the subjects had to name a set of new messages that had been added to the original version of the diagrams (representing additional required behavior for the sequence diagram). Only the message parameters were provided. This task was used to calculate *TransEffec* and *TransEffic*.

4.2. The operation of the experiments

Before each of the experiments took place, the subjects attended a seminar in which the stereotypes under study and their use were explained. It was the first contact that all the subjects had with these stereotypes and the seminar speaker and materials were exactly the same in the experiment and the two replications. As well as all this, the subjects were shown an example, similar to the material used in the experiment and were told how to solve the tasks in that example.

Each of the experiments consisted of two rounds. In each round, each of the groups was given a different treatment. As previously mentioned, in order to alleviate learning effect, we divided the experimental subjects into four groups. We assigned the corresponding diagrams to each group at random, but gave them out in a different order in each case. Table 1 presents the outline of the experimental operation. The description of each type of diagram has already been presented in the previous section. This assignment of diagrams corresponds with the selected balanced factorial design with group-interaction confounding [32], which permits the lessening of the effects of learning and fatigue.

Table 1
Experiment rounds.

Round 1		Diagram type	
		S	*N*
Domain	A	Group 1	Group 2
	B	Group 4	Group 3
Round 2		Diagram type	
		S	*N*
Domain	A	Group 3	Group 4
	B	Group 2	Group 1

4.3. The conducting of the experiments

The experiment and replications took place in a one and a half hour session. In the 30 first minutes we explained how to perform the experiment and randomly assigned the subjects to four different groups. Within these groups, there were two pairs that received the same models but in different order, so that the possible learning effect could be alleviated. In this way, we worked with two balanced groups.

They were conducted in a classroom, where the students were supervised and no communication among them was allowed. Both groups were located in the same room.

Each round (see Table 1) was performed in the following way:

- The subjects received the material for Test 1, which included a UML sequence diagram and a questionnaire. After finishing this task, Test 1 was handed back to the supervisor.
- The students received Test 2, which had to be solved without the UML sequence diagram. After completing the tasks, Test 2 was handed back to the supervisor.
- The students received Test 3 and after solving this they returned it to the supervisor, again without the UML sequence diagram.

4.4. Data analysis and interpretation

In this section, we present, for each variable, the descriptive statistics and the results of the Kruskal–Wallis tests carried out to test the formulated hypotheses in the different studies. Kruskal–Wallis is the most appropriate test with which to explore the results of a factorial design with interaction confounded [32,39], i.e., the design used in our experiments, when there is non-normal distribution of the data. All the statistical analyses presented in this section were performed using SPSS [40].

Table 2
Descriptive statistics for *SCEffec and SCEffic.*

Diag. type	*SCEffec*						*SCEffic*					
	EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30		EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30	
	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD
Stereot.	**0.772**	0.14	**0.768**	0.16	**0.827**	0.10	**0.237**	0.11	**0.236**	0.09	**0.316**	0.13
Non stereot	0.755	0.11	0.728	0.13	0.793	0.15	0.235	0.09	0.236	0.10	0.290	0.10

Table 2 presents the descriptive statistics for the effectiveness and efficiency for all the variables studied: Semantic Comprehension, Retention and Transfer. Cells in bold indicate the best subjects' performance in each one of the tests.

At a glance, we can observe that when the subjects used stereotypes they obtained better values in 5 out of 6 variables.

More specifically, the tests related to *SCEffec* and *SCEffic* were always performed better when the diagrams contained stereotypes. This indicates that stereotypes may improve the comprehension of the sequence diagrams.

Retention variable shows a conflicting trend. *RetenEffec* measures how good the retention of the subjects is, independently of the time taken: the time is taken into account with *RetenEffic*. The results show that the subjects obtained better results in *RetenEffec* when they did not use stereotypes, while the scores were higher for *RetenEffic* when they worked with stereotyped diagrams. Therefore, the subjects who used non-stereotyped diagrams were able to recall a greater number of correct functionalities of the models, but the subjects with non-stereotyped diagrams performed better if we related the number of correct answers to the time spent answering the test. This is probably due to the fact that stereotyped diagrams contain more information than those which are non-stereotyped and thus prove to be more difficult to remember (not to understand!). The higher *RetenEffic* for the stereotyped diagrams seems to confirm that the students answered what they remembered in a short time and did not consider the rest of answers Table 3.

Finally, as regards *Transfer* (Table 4), for both Effectiveness and Efficiency the descriptive statistics point to better results in 2 out of 3 experiments when the subjects used the stereotyped diagrams. Unfortunately, at this time, we have no explanation for the contradictory data (0.513 and 0.567).

We have tested the hypothesis formulated with a Kruskal–Wallis, obtaining the results shown in Table 5. Bolded values marked with an asterisk (*) indicate a significance level below 0.05.

Analyzing the results shown in Table 5, we can argue that:

- The *Ster* variable (related to the use or not of stereotypes), on its own, does not seem to significantly affect any of the independent variables. However, when it is related to the domain it always affects the retention and, in most cases, the transfer too. This can indicate that using stereotypes improves the comprehension of the models especially in those domains that are not completely well-known.
- The domain definitely affects all the variables and so does the use of stereotypes when we evaluate its interaction with the domain of the diagrams. This can be explained because one of the domains was more difficult than the other.

So, as a conclusion from this data analysis and considering both the descriptive statistics and the hypothesis testing carried out, we can observe that the stereotypes presented in this work, as confirmed by descriptive statistics, improve the comprehension of UML sequence diagrams but on the negative side, this result lacks strong statistically significant evidence.

As scientists, we cannot reject the null hypotheses investigated in this work and accept the alternative hypotheses (this could be risky and misleading, in that these results might be a consequence of the treatment or be determined by an interaction effect between treatment and domain). As practitioners, however, by basing our judgement only on the descriptive statistics we can conclude that there is a tendency for stereotypes to improve, to some extent, the three perspectives of comprehension according to the CTML [36]: Semantic Comprehension, Retention and Transfer, especially in domains that are not very familiar ones.

The effect of the domain seems reasonable, for when it comes to understanding sequence diagrams, if the subjects are not familiar with the domain; they rely more heavily on stereotypes. That is because they are familiar with the semantics of these, even if they may not be totally acquainted with the semantics of the domain.

Table 3
Descriptive statistics for *RetenEffec and RetenEffic.*

Diag. type	*RetenEffec*						*RetenEffic*					
	EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30		EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30	
	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD
Stereot.	0.706	0.19	0.794	0.16	0.661	0.21	**0.282**	0.14	**0.332**	0.17	**0.353**	0.16
Non stereot.	**0.721**	0.18	**0.806**	0.14	**0.669**	0.17	0.274	0.11	0.326	0.14	0.339	0.14

Table 4
Descriptive statistics for *TransEffec and TransEffic.*

Diag. type	*TransEffec*						*TransEffic*					
	EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30		EXP *N* = 69		REP1 *N* = 25		REP2 *N* = 30	
	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD
Stereot.	**0.758**	0.26	**0.896**	0.19	0.503	0.31	0.547	0.48	**0.357**	0.16	**0.390**	0.30
Non stereot.	0.722	0.28	0.736	0.23	**0.513**	0.31	**0.567**	0.50	0.340	0.24	0.323	0.26

Table 5
Kruskal–Wallis tests results.

		SCEffec	SCEffic	RetenEffec	RetenEffic	TransEffec	TransEffic
EXP	Ster	0.354	0.969	0.718	0.920	0.468	0.841
	Dom	0.847	**0.019***	**<0.001***	**<0.001***	**<0.001***	**<0.001***
	Ster*Dom	0.458	0.068	**<0.001***	**<0.001***	**<0.001***	**<0.001***
REP1	Ster	0.155	0.756	0.961	0.786	**0.004***	0.352
	Dom	0.960	0.295	**<0.001***	**<0.001***	**0.002***	**<0.001***
	Ster*Dom	0.547	0.668	**<0.001***	**0.005***	**<0.001***	**<0.001***
REP2	Ster	0.486	0.918	0.900	0.564	0.940	0.437
	Dom	0.300	0.114	**<0.001***	**<0.001***	0.747	0.801
	Ster*Dom	0.664	0.472	**0.001***	**0.001***	0.987	0.874
ALL	Ster	0.072	0.812	0.823	0.935	0.131	0.366
	Dom	0.926	**0.004***	**<0.001***	**<0.001***	**<0.001***	**<0.001***
	Ster*Dom	0.281	**0.031***	**<0.001***	**<0.001***	**0.001***	**<0.001***

Although these results are encouraging with respect to the use of stereotypes, we are aware that more experimentation is needed if more conclusive results are to be obtained.

4.5. Threats to validity

We must consider certain issues which may have threatened the validity of the experiment:

4.5.1. External validity

External validity may be threatened when experiments are performed with students, doubting the representativeness of the subjects with respect to software professionals. In spite of this, the tasks to be performed did not require high levels of industrial experience, so we believed that this experiment could be considered appropriate, as suggested in the literature [28]. Our subjects have enough knowledge to perform the required tasks. Also related to external validity, even the original materials of the experiments used UML 1.7, which was the most used standard notation in that moment; we consider that the obtained results are still valid for UML 2 sequence diagrams since changes in the new version of the language do not affect the elements related to the proposed stereotypes. Therefore, both the stereotypes and the results of this study are equally valid in both versions of UML. The stereotypes were proposed to express the different types of possible messages in sequence diagrams. As there is nothing new in UML 2 that deserves a new type of message, in our opinion the proposed stereotypes can be used exactly the same way in UML 2 sequence diagrams. Besides, we also decided to update the version of the diagrams used because the materials are ready in case anyone in the community wanted to perform an external replication. Finally, a threat that might affect the external validity concerns the size and complexity of the sequence diagrams used. In order to avoid biasing the results, we decided to use relatively small tasks since a controlled experiment requires that participants complete the assigned tasks in a limited amount of time. In all the experiments, the participants had 1 h to complete three tests (each one with a UML sequence diagram and a questionnaire attached), thus constraining their size and complexity. In all the cases the requirements represented by the sequence diagrams used in the tasks have been selected from actual software systems.

4.5.2. Internal validity

Internal validity threats are mitigated by the design of the experiment. Each group of subjects worked on the domain in different orders. Nevertheless, there is still the risk that the subjects might have learned how to improve their performances from one performance to the second one.

4.5.3. Conclusion validity

The random heterogeneity of subjects is always present when experimenting with students and we are also conscious that they had no previous knowledge of the stereotypes used in the UML sequence diagram included in the experimental materials. Furthermore, if the knowledge of the students involved in the experiment could be assumed to be comparable to that of junior industry professionals, the working pressure and the overall environment in industry is different. We assumed a homogenous background in the subjects, but this was not evaluated before the experiment was carried out.

4.5.4. Construct validity

The hypothesis of equivalence between the complexities of the domains was not confirmed by data analysis and this has been one of the pitfalls of this work. In fact, there were relevant differences in the subject performances which can be attributed to the differences in the diagram domains: the car rental diagram was found to be much more difficult than that of the hotel.

5. Meta-analysis study

Although we have obtained and presented significant results which showed that the use of stereotypes improves the way that subjects comprehend the diagrams, we have decided to integrate the results of the different studies, in order to obtain stronger results.

There are several statistical methods that allow us to accumulate and interpret a set of results obtained through different experiments that are inter-related because they check similar hypotheses [41–45]. In the present study, we have used meta-analysis because it allows us to extract more general conclusions, as the experimental conditions were the same.

Meta-analysis is a set of statistical techniques for combining the different effect sizes of the experiments to obtain a global effect of a factor. As measures may come from different environments and not be homogeneous, a standardized measure of each one needs to be obtained and then those measures for estimating the global size effect of the factor must be combined. In our study, the factor is the use of stereotypes and how that affects the comprehension of UML sequence diagrams.

To carry out the meta-analysis presented in this work we used the Meta-Analysis v2 tool [46]. In this meta-analysis, for each variable we used the mean value for the diagrams with stereotypes minus the mean value for the diagrams without stereotypes, and from these values we obtained Hedges' g metric [42,47], which we used as standardized measure. This value expresses the magnitude of the treatment effects- in our case the use of stereotypes-relative to the within-group standard deviations.

Table 6
Hedge's g metric values.

	SCEffec	SCEffic	RetenEffec	RetenEffic	TransEffec	TransEffic
Global effect size	0.193 (*S*)	0.065 (*S*)	−0.071 (*S*)	0.065 (*S*)	0.211 (*S*)	0.051 (*S*)

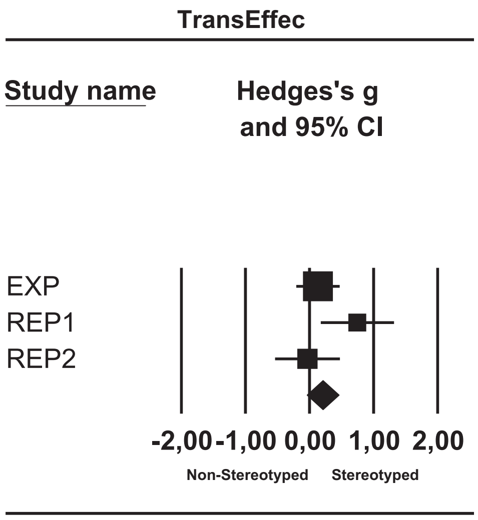


Fig. 4. *TransEffec* meta-analysis results.

The Hedges' g metric is a weighted mean whose weights depend on sample size (see Eq. (1))

$$\overline{Z} = \frac{\sum_i w_i z_i}{\sum_i w_i} \quad (1)$$

where $w_i = 1/(n_i - 3)$ and n_i is the sample size of the ith experiment.

The higher the value of Hedges' *g* is, the higher the corresponding mean difference is too. For studies in Software Engineering, we can classify effect sizes into three different values: small (*S* – 0–0.37), medium (*M* – 0.38–1.00) and large (*L* – above 1.00) [47]. Table 6 summarizes the results we obtained in our meta-analysis. For instance, an effect size of 0.5 indicates that the mean value obtained when using it is half a standard deviation larger than the mean when not using it. A positive value means that using stereotypes improves the part of the comprehension measured by that variable.

By way of example, we also present one of the meta-analysis results in diagram form, as provided by the Meta-Analysis v2 tool [46]. Fig. 4 displays Hedges'g metric with a confidence interval of 95% for the *TransEffec* variable. Not all the studies contribute equally to the overall conclusion, which is represented by the diamond in the last row of the figures. Each of them receives a specific weight in the meta-analysis, i.e., the study's effect size, represented by the squares in the figures. The estimations for studies with a large sample size are more accurate, so they contribute more to the overall effect. However, sample size is not the only factor contributing to the weight of a study. The weight of a study is proportional to the area of the corresponding square in the figures.

The effect size obtained is small in all cases, probably because of the scarce number of studies used in the data meta-analysis. Nevertheless, as we can see in Table 6 and Fig. 4, the results are consistent with those previously presented and all the variables (except for *RetenEffic*) present a positive effect, i.e. using stereotypes helps to improve the semantic comprehension, transfer and retention of UML sequence diagrams, so the meta-analysis strengthens the conclusions commented on previously.

6. Conclusions and future work

The main concern of this paper is to investigate the use of stereotypes in the context of UML sequence diagrams, from three different perspectives of the comprehension of these diagrams: Semantic Comprehension, Retention, and Transfer. This was done through a family of experiments, consisting of a controlled experiment and two replications of this.

The results obtained are twofold:

- On one hand, we have concluded that using the stereotypes that we have presented in this work improves the comprehension of UML sequence diagrams, especially when the domain of the system that is being modeled is not really well-known. More specifically, considering the CTML [36], and the results obtained in the data analysis and meta-analysis, we can conclude that using these stereotypes improves.
  - The ability to comprehend the semantics of the models (*Semantic Comprehension*).
  - The ability to retain knowledge from it (*Retention*).
  - The ability to use the knowledge gained from the material to solve related problems which are not directly answerable from it (*Transfer*).
- On the other hand, the magnitude of the results is not statistically significant enough to be generalized. Actually, the hypothesis tests carried out and the meta-analyses done do show a positive effect of the proposed stereotypes but, at the same time, the size of this effect is too small to assume these results to be definitive.

So, although the results show a tendency towards stereotypes bringing improvement to the understanding of sequence diagrams, further experimentation is needed to reinforce the results. We are planning to carry out new experiments with practitioners instead of with students, using larger and more complex diagrams and considering two domains in advance. One of these will be well-known by the subjects, but they will not be familiar with the other, which means we will have one domain that is easier to understand than the other. This would allow us to clarify the interaction effect observed between domain and stereotypes, as well as to reach a definitive clarification of the results obtained.

Comparing these results with those obtained in previous related works [2,4], we can observe how they are all in concordance and the use of stereotypes helps improving the comprehension of the models.

Although further research is necessary for strengthening these results, introducing these stereotypes both in academia and industry could be an interesting practice for checking the validity of the results.

Acknowledgements

This research is part of the following projects: "MEDUSAS (IDI-20090557)" and "Proyecto Integrado ORIGIN (IDI-2010043(1-5))" financed by CDTI and FEDER; "ECCOO (TRA2009_0074)", "MAGO/PEGASO (TIN2009-13718)" and "MULTIPLE (TIN2009-13838)" financed by MICINN; MECCA (PII2I09-0075-8394) and IMPACTUM

(PEII11-0330-4414) financed by JCCM and CALIMO funded by the Generalitat Valenciana.

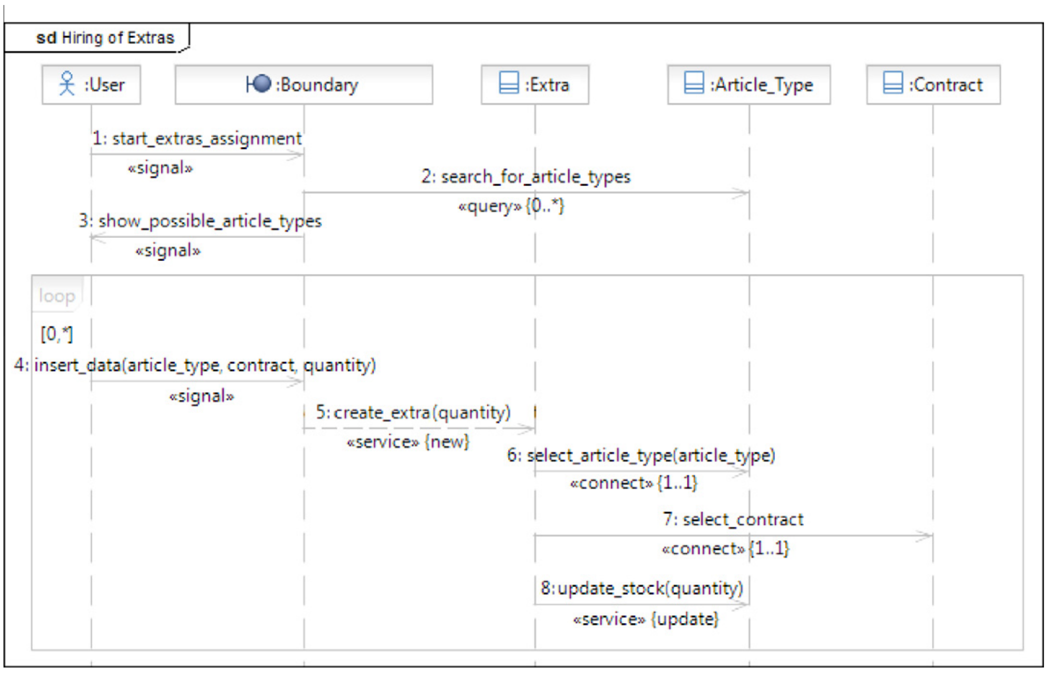
Appendix A

Next, as an example, we present the materials related to the rent-a-car domain. For the reader's convenience, the original materials have been translated into English.

A.1. Hiring of extras

This sequence diagram represents the hire of extras associated with a car-rental contract. Each car hire contract has a set of extras associated with it. For example, the hire of a mobile telephone, child's seat, etc. In the diagram the object, the object: ArticleType represents a specific type of article of which there may be many available for renting.

For example, there may be many GPS available for renting in the firm.


sd Hiring of Extras
:User
:Boundary
:Extra
:Article_Type
:Contract
1: start_extras_assignment
«signal»
2: search_for_article_types
«query» {0..*}
3: show_possible_article_types
«signal»
loop
[0,*]
4: insert_data(article_type, contract, quantity)
«signal»
5: create_extra(quantity)
«service» {new}
6: select_article_type(article_type)
«connect» {1..1}
7: select_contract
«connect» {1..1}
8: update_stock(quantity)
«service» {update}

WRITE DOWN THE PRESENT TIME(**HH:MM:SS**)

Answer the following Yes/No type questions:

1. Can a rental contract have various extras assigned to it? _______
2. Is it necessary to create a new object to rent an extra? _______
3. Must every extra of a rental contract be associated to a car rental contract? _______
4. Is it the system that identifies the contract with which the rental of an extra is related? _______
5. Would it be possible to hire three child seats for a car rental contract? _______
6. Does the type of car hired have any effect in the search for types of articles to rent? _______
7. Is it possible to up-date the stock to register the hiring of more items out of those available? _______
8. When an extra is rented, is a new type of car hire contract created? ________
9. Does the search for types of articles allow us to know all the kinds of articles that may be hired? _______
10. Is it possible to assign the hiring of an article not in stock to a car-rental contract? _______

WRITE DOWN THE PRESENT TIME (HH:MM:SS)

A.2. Hiring of extras (2)

WRITE DOWN THE PRESENT TIME (HH:MM:SS)

Fill in the blanks in the following text to describe the functionality of the diagram:

The diagram represents the rental of extras associated with a _______ of a car. Each may have a _______ associated with it: hire of a mobile telephone, child SEAT, GPS, etc.

In the diagram, the object: *ArticleType* represents a specific _______, for example a GPS, of which there may be _______ to rent.

The process of rental starts when the user begins the _______ of extras. The interface then carries out a _______ to obtain the types of _______ which can be hired and these are shown back to the user.

Later, and as long as articles are wished to be added, the process set out below is carried out.

Firstly _______ enters the data are to the system, namely the kind of article, the quantity of items that have been chosen and the _______ contract to which the item will be assigned.

All this information generates a new instance of a _______ object to which the information set out above is sent. Then, the chosen _______ is selected and its _______ is modified. Apart from this, the rental extra created is to the corresponding car hire contract.

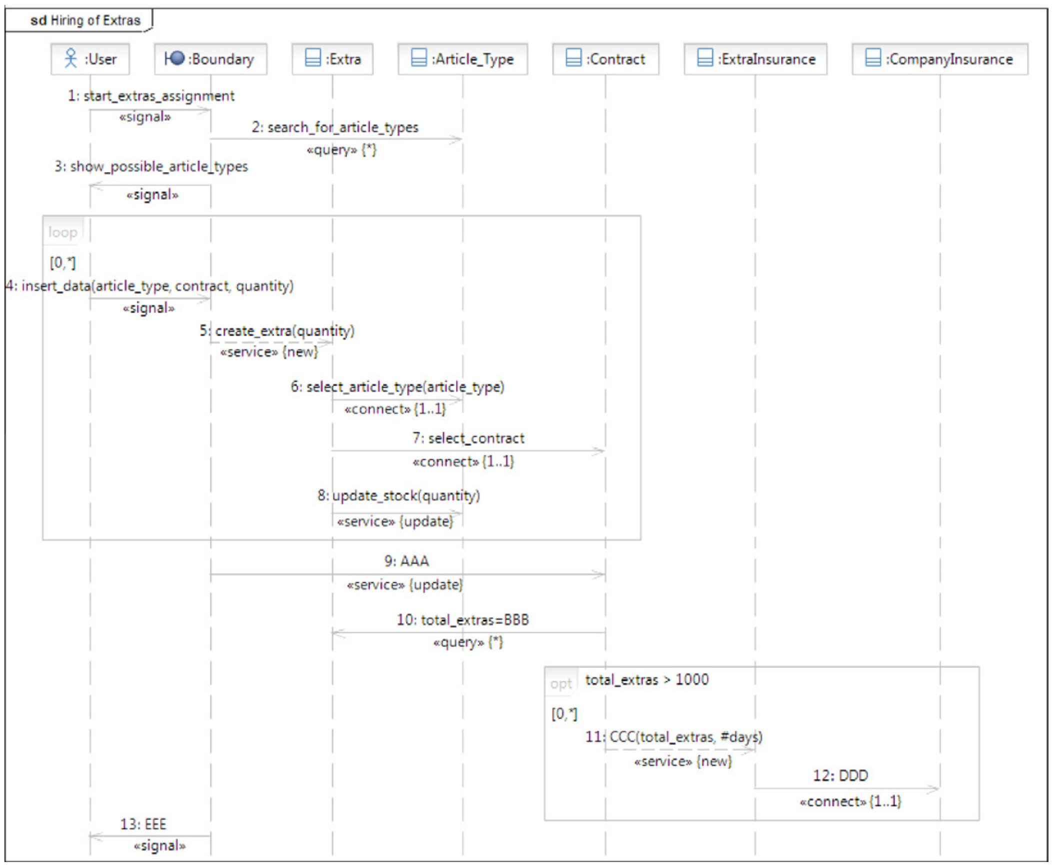
WRITE DOWN THE PRESENT TIME (HH:MM:SS) _______

A.3. Hiring of extras (3)

When all the car-hire extras have finally been entered, the total amount of extras should be worked out. If the total amount of extras comes to an amount exceeding a maximum (1000 euros), an insurance policy should be created for these extras, taking into account the amount of these and the number of days of rental. This policy should be assigned to an insurance company. A receipt of the details of all this operation is shown at the end.

References

[1] OMG, UML 2.2 Unified Modeling Language, TM, 2009.
[2] M. Staron, L. Kuzniarz, C. Wohlin, Empirical assessment of using stereotypes to improve comprehension of UML models: a set of experiments, The Journal of Systems and Software 79 (2006) 727–742.
[3] J. Conallen, Building Web Applications with UML, Addison-Wesley Publishing Company, Reading, USA, 2000.
[4] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, M. Ceccato, Developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: a series of four how experiments, IEEE Transactions on Software Engineering 36 (1) (2010) 96–118.
[5] M. Genero, M.E. Manso, A. Visaggio, G. Canfora, M. Piattini, Building measure-based prediction models for UML class diagram maintainability, Empirical Software Engineering 12 (2007) 517–549.
[6] M.C. Otero, J.J. Dolado, Evaluation of the comprehension of the dynamic modeling in UML, Information and Software Technology 46 (1) (2004) 35–53.



AAA: _______	BBB: _______
CCC: _______	DDD: _______
EEE: _______	

Please give appropriate names to the messages in the diagram:

WRITE DOWN THE PRESENT TIME (HH:MM:SS)

WRITE DOWN THE PRESENT TIME (HH:MM:SS)

[7] H. Reinhartz-Berger, D. Dori, OPM vs. UML – experimenting with comprehension and construction of web application models, Empirical Software Engineering 10 (2005) 57–79.
[8] B. Selic, The pragmatics of model-driven development, IEEE Software 20 (5) (2003) 19–25.

[9] E. Insfran, P. Pastor, R. Wieringa, Requirements engineering-based conceptual modeling, Journal of Requirements Engineering 7 (2) (2002) 61–72.

[10] E. Insfran, A requirements engineering approach for object-oriented conceptual modeling, in: DSIC, University of Technology of Valencia, 2003.

[11] E. Arisholm, L. Briand, S. Hove, Y. Labiche, The impact of UCLM documentation on software maintenance: an experimental evaluation, IEEE Transactions on Software Engineering 32 (6) (2006) 365–381.

[12] J. Miller, Applying meta-analytical procedures to software engineering experiments, Journal of Systems and Software 54 (2000) 29–39.

[13] L.M. Pickard, Combining empirical results in software engineering, University of Keele, 2004.

[14] J.A. Cruz-Lemus, M. Genero, M.E. 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.

[15] H.C. Purchase, L. Colpoys, M. McGill, D. Carrington, C. Britton, UML class diagram syntax: an empirical study of comprehension, in: Australian Symposium on Information Visualisation, Sydney, Australia, 2001.

[16] H.C. Purchase, L. Colpoys, M. McGill, D. Carrington, UML collaboration diagram syntax: an empirical study of comprehension, in: 1st International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT'02), Paris, France, 2002.

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

[18] C.F.J. Lange, M.R.V. Chaudron, Interactive views to improve the comprehension of UML models – an experimental validation, in: 15th IEEE International Conference on Program Comprehension (ICPC'07), Banff, Canada, 2007.

[19] S. Xie, E. Kraemer, R.E.K. Stirewalt, Empirical evaluation of a UML sequence diagram with adornments to support understanding of thread interactions, in: 15th IEEE International Conference on Program Comprehension (ICPC'07), Banff, Canada, 2007.

[20] S. Yusuf, H. Kagdi, J.I. Maletic, Assessing the comprehension of UML class diagrams via eye tracking, in: 15th IEEE International Conference on Program Comprehension (ICPC'07), Banff, Canada, 2007.

[21] M. Staron, L. Kuzniarz, C. Thurn, An empirical assessment of using stereotypes to improve reading techniques in software inspections, in: 3-WoSQ: Third Workshop on Software Quality, ACM, St. Louis, USA, 2005.

[22] M. Genero, J.A. Cruz-Lemus, D. Caivano, S. Abrahao, E. Insfran, J.A. Carsí, Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: a controlled experiment, in: 11th ACM/IEEE MODELS Conference, LNCS 5301, 2008.

[23] I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard, Object-Oriented Software Engineering, A Use Case-driven Approach. Reading, Addison-Wesley, USA, 1992.

[24] H.E. Eriksson, M. Penker, UML Toolkit, John Wiley and Sons, 1998.

[25] C. Larman, Applying UML and Patterns, Prentice-Hall, 1998.

[26] A.V. Lamsweerde, Inferring declarative requirements specifications from operational scenarios, IEEE Transactions on Software Engineering 24 (12) (1998) 1089–1114.

[27] S. Berner, M. Glinz, S. Joos, A classifiction of stereotypes for object-oriented modeling languages, in: 2nd International Conference on the Unified Modeling Language, Springer, Fort Collings, USA, 1999.

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

[29] V. Basili, D. Weiss, A methodology for collecting valid software engineering data, IEEE Transactions on Software Engineering 10 (6) (1984) 728–738.

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

[31] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publisher, 2000.

[32] R.E. Kirk, Experimental Design: Procedures for the Behavioral Sciences, Brooks/Cole Publishing Company, 1995.

[33] J.A. Cruz-Lemus, M. Genero, S. Morasca, M. Piattini, Using practitioners for assessing the understandability of UML Statechart diagrams with composite states, Lecture Notes on computer Science 4802 (2007) 213–222.

[34] A. Gemino, Y. Wand, Complexity and clarity in conceptual modeling: comparison of mandatory and optional properties, Data and Knowledge Engineering 55 (2005) 301–326.

[35] F. Bodart, A. Patel, M. Sim, R. Weber, Should optimal properties be used in conceptual modelling? A theory and three empirical tests, Information Systems Research 12 (4) (2001) 384–405.

[36] R.E. Mayer, Multimedia Learning, Cambridge University Press, 2001.

[37] R.E. Mayer, Models for understanding, Review of Educational Research 59 (1) (1989) 43–64.

[38] V. Basili, H. Rombach, The TAME project: towards improvement-oriented software environments, IEEE Transactions on Software Engineering 14 (6) (1988) 758–773.

[39] B.J. Winer, D.R. Brown, K.M. Michels, Statistical Principles in Experimental Design, McGraw-Hill, 1991.

[40] SPSS, SPSS 12.0, Syntax Reference Guide, SPSS Inc., Chicago, USA, 2003.

[41] G.V. Glass, B. McGaw, M.L. Smith, Meta-Analysis in Social Research, Sage Publications, 1981.

[42] L.V. Hedges, I. Olkin, Statistical Methods for Meta-Analysis, Academia Press, 1985.

[43] R. Rosenthal, Meta-Analytic Procedures for Social Research, Sage Publications, 1986.

[44] J.A. Sutton, R.K. Abrams, R.D. Jones, A.T. Sheldon, F. Song, Methods for Meta-Analysis in Medical Research, John-Wiley & Sons, 2001.

[45] F.M. Wolf, Meta-Analysis: Quantitative Methods for Research Synthesis, Sage Publications, 1986.

[46] Biostat, Comprehensive Meta-Analysis v2, 2006.

[47] V. Kampenes, T. Dybå, J.E. Hannay, D.I.K. Sjoberg, A systematic review of effect size in software engineering experiments, Information and Software Technology 49 (11–12) (2007) 1073–1086.