The Role of Experience and Ability in Comprehension Tasks supported by UML Stereotypes

Filippo Ricca[1], Massimiliano Di Penta[2], Marco Torchiano[3], Paolo Tonella[1], Mariano Ceccato[1]

[1]ITC-irst, Trento, Italy

[2]RCOST - University of Sannio, Benevento, Italy

[3]Politecnico di Torino, Italy

ricca@itc.it, dipenta@unisannio.it, torchiano@polito.it, tonella@itc.it, ceccato@itc.it

Abstract

Proponents of design notations tailored for specific application domains or reference architectures, often available in the form of UML stereotypes, motivate them by improved understandability and modifiability. However, empirical studies that tested such claims report contradictory results, where the most intuitive notations are not always the best performing ones. This indicates the possible existence of relevant influencing factors, other than the design notation itself.

In this work we report the results of a family of three experiments performed at different locations and with different subjects, in which we assessed the effectiveness of UML stereotypes for Web design in support to comprehension tasks. Replications with different subjects allowed us to investigate whether subjects' ability and experience play any role in the comprehension of stereotyped diagrams. We observed different behaviors of users with different degrees of ability and experience, which suggests alternative comprehension strategies of (and tool support for) different categories of users.

1. Introduction

General purpose design notations may be inadequate to model the features of specific application domains and system architectures. For example, Web applications are organized around their navigation structure, which is hard to model using standard design notations, such as basic (i.e., non-stereotyped) UML [17]. Special purpose or extended design notations (such as UML stereotypes) promise a more intuitive and direct mapping between the modeled entities and the implementation. However, they require that users become familiar with a less common and less standardized design language, often containing domain-specific

terms/icons. Users (or some categories of users) might prefer to derive the mapping from the standard sources of information (e.g., code and basic UML), avoiding the details of the specialized notation.

In this paper we investigate the reaction of different categories of users, when they are given the possibility to access special purpose design notations. We discriminate users according to the respective level of experience and ability, with the purpose of testing the hypothesis that these are relevant influencing factors that should be taken into account when adopting such kinds of notations. Specifically, we consider design notations that have been proposed to support the development of Web applications. Among the most referenced approaches are WebML [6], WSDM [9], OOHDM [18], and WAE [7], many of which are extensions of UML. We evaluated the effectiveness of Conallen's stereotypes (WAE — Web Application Extension — notation [7]) in improving the comprehension of Web applications. Conallen's stereotypes are focused on the navigation structure of a Web application, with a straightforward mapping into the implementation. Although just one example of UML stereotypes, it is quite representative of those extensions that aim at covering domain specific concepts which are missing in the basic notation.

We have designed an empirical study where different groups of subjects perform comprehension tasks on Web applications for which design notations are available in addition to the source code. Either non-stereotyped or Conallen's UML diagrams were provided to the subjects, who were free to use them or not, as in a real program comprehension setting. The experimentation consisted of a family of three experiments, two performed at the University of Trento, Italy, and one performed at the University of Sannio, Benevento, Italy, with three different groups of subjects. In Trento the experiment was run both with undergraduate and with graduate students; in Benevento it was run with graduate students. The level of ability of the

involved subjects was ranked according to the scores they obtained in previous courses. Results indicate that the usage 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.

While enriched design notations and UML stereotypes have already been evaluated empirically in previous works [2, 15, 19], this is the first attempt to investigate in detail the role of experience and ability, as relevant influencing factors that add to the design notation itself. To this aim, the study comprised three replications made with different subjects. Results obtained in one of the replications have already been presented elsewhere [15], but it is only when data from all three studies have become available that we could carry out analyses to evaluate the role of experience and ability.

The paper is organized as follows: Section 2 describes the design of the empirical study that we conducted. Results are presented in Section 3 and discussed in Section 4, followed by the threats to validity (Section 5). Conclusions and future works are given in Section 7, after comparing the present study with related works (Section 6).

2. Experimental design

This section describes the definition, design and settings of the proposed experimentation following the guidelines by Wohlin *et al.* [22] and Juristo and Moreno [11]. Table 1 summarizes the main elements of the experimentation.

2.1 Experiment definition and context selection

The *goal* of the study is to analyze the use of stereotyped UML diagrams, with the *purpose* of evaluating their usefulness in Web application comprehension for different categories of users. The *quality focus* is ensuring high comprehensibility and maintainability, while the *perspective* is both of *Researchers*, evaluating how effective are the stereotyped diagrams during maintenance for different categories of users, and of *Project managers*, evaluating the possibility of adopting a Web application design and reverse engineering tool in her/his organization, depending on the skills

Table 1. Overview of the experiment

Goal	Analyze the support given by Conallen's stereotypes on comprehension tasks and the the influence of subjects' ability and experience.
Context	Diagrams (basic UML and Conallen's) reverse engineered from the code.
Null hypothesis	No effect on comprehension.
Main factor	Design notation used: basic UML (UML) vs. stereotypes (Conallen).
Other factors	Subjects' Experience and Ability, Systems.
Dependent variables	Comprehension level.

Table 2. Characteristics (a) of the systems under study and (b) of the experimental subjects.

Claros			WfMS		
	Files	LOC		Files	LOC
Java	44	6288	Java	85	2378
JSP	34	1996	JSP	7	431
Total	78	8284	**Total**	92	2809

(a)

Experiment	Location	Course type	# of students
Exp I	Trento	Graduate	13
Exp II	Trento	Undergraduate	35
Exp III	Benevento	Graduate	18

(b)

of the involved developers. The *context* of the experiment consists of *objects*, two Web applications and of *subjects*, students from three courses, an undergraduate course and two graduate courses respectively.

The experimentation objects are two Java-based Web applications, *Claros*[1] and *WfMS*[2] [4]. Both are small/medium size open source applications (see Table 2-a) based on the Servlet/JSP technology and downloaded from the Internet; they are small enough to fit the time constraint of the experimental sessions. Although commercial or institutional Web applications may be larger, the application domains of the selected systems is pretty typical of existing Web applications. *Claros* is an on-line Web mail management application. *WfMS* is a simple workflow management system that allows the definition of processes and their enactment. While WfMS is larger in terms of classes, Claros has a larger design view and a more complex navigational model. For the two systems, both the source code and UML class diagrams — drawn with and without Conallen's stereotypes [7] — were available. Both applications were

[1]http://www.claros.org
[2]http://www.pearsoned.co.uk/HigherEducation/Booksby/ BrugaliTorchiano/

designed using the Model-View-Controller (MVC) pattern and the View, on which stereotypes were used, comprises 19 UML classes (38 in the Conallen's diagram) for Claros and 13 (24 in the Conallen's diagram) for WfMS. Diagrams (limited to the View) are shown in a technical report [16].

The study was executed twice at the University of Trento, with different subjects, and once at the University of Sannio. The subjects participating in the two replications in Trento are 13 Master students (2nd year M.Sc.) attending the Laboratory of Software Analysis and Testing (Exp I), and 35 Bachelor students (2nd year B.Sc.) attending the Laboratory of Software Engineering course (Exp II). At the University of Sannio, the 18 Master students (1st year M.Sc.) attending the course on development of Web-based systems (Exp III) were involved. Information about the subjects taking part to each experiment is summarized in Table 2-b. Within each replication, all the students are from the same class with, roughly, the same background. Bachelor students had attended previously Programming and Software Engineering courses (which is of course true also of Master students). All subjects had a fairly good knowledge of UML and Java.

2.2 Hypotheses formulation and variable selection

The main objective of our study is to investigate the effect of UML stereotypes on software comprehension. We assume that stereotypes might have a positive effect, thus we formulate a one-tailed null hypothesis and the related alternative hypothesis:

H_0 When performing a comprehension task the use of stereotyped class diagrams (versus non-stereotyped class diagrams) **does not significantly improve** the comprehension level achieved by maintainers.

H_a When performing a comprehension task the use of stereotyped class diagrams (versus non-stereotyped class diagrams) **significantly improves** the comprehension level achieved by maintainers.

Also, our study is devoted to investigating how subjects' experience (graduate vs. undergraduate students) and ability interact with the use of stereotypes and affect the achieved comprehension level. This requires us to formulate three further null hypotheses, this time two-tailed, since it cannot be guessed whether the ability or experience have a positive or negative effect:

H_{0e} Subjects' experience **does not significantly interact** with the use of stereotyped class diagrams to influence the comprehension level achieved by maintainers.

H_{0a} Subjects' ability **does not significantly interact** with the use of stereotyped class diagrams to influence the comprehension level achieved by maintainers.

H_{0ea} Subjects' ability and experience **do not significantly interact** with the use of stereotyped class diagrams to influence the comprehension level achieved by maintainers.

The related alternative hypotheses are easily derived from the one given above for the main factor. The main factor (hereby referred as *Method*) of this experimentation is the use of UML stereotypes, in particular of Conallen's stereotypes for Web application modeling [7]. Since this notation extends UML, the notation used for comparison (control group) is basic UML, with no Web-specific stereotype. Thus, the *Method* factor can assume one of the values in {*UML, Conallen*}. As is often the case, a thorough UML documentation is not available, therefore diagrams have been reverse engineered from the code and then properly adjusted, so as to reproduce a situation where diagrams are aligned with the code and at the same time represent a meaningful and compact abstraction of the implementation.

Other than the *Method*, the experimental hypotheses are defined in terms of two other factors, *Experience* and *Ability*. Regarding the *Experience*, Exp I and Exp III subjects were classified as *Graduate (G)*, while Exp II subjects as *Undergraduate (U)*. A quantitative assessment of the *Ability* level of each involved subject was obtained by resorting to the average grades obtained at the previous exams. Subjects with average grades below a fixed threshold were classified as *low (l) Ability*, and the remaining ones as *high (h) Ability*. Finally, other experimental factors are the *System* (Claros or WfMS) and, because of the experimental design (see Section 2.3), the experimental session in which a task was performed (*Lab 1* or *Lab 2*).

The main outcome observed in the study is the *comprehension level*. To evaluate it, we asked the subjects to answer a questionnaire and assessed the answers using an Information Retrieval approach. Since the answers to each question consists of a list of system elements, i.e. classes, JSPs, HTML pages, we can count:

$A_{s,i}$ set of elements mentioned in the answer to question *i* by subject *s*; and

C_i the correct set for elements expected for the question *i*.

Based on the above definition, we computed *precision* and *recall* for each answer [10]. Precision measures the fraction of items in the answer that are correct:

$$precision_{s,i} = \frac{|A_{s,i} \cap C_i|}{|A_{s,i}|}$$

Table 3. Sample questions (5 out of 12) for WfMS.

ID	Question
1	Suppose that you have to set the background color of each Web page using CSS (Cascading Style Sheets). Which classes/pages does this change impact?
2	Suppose that you have to substitute, in the entire application, the form-based communication mechanism between pages with another mechanism (i.e. Applet, ActiveX, ...). Which classes/pages does this change impact?
3	Does the application conform to the Model-View-Controller (MVC) pattern? If yes which class (or classes) implements the controller component?
4	The description of a process is made up of three main types of elements (activity, participant, and transition) and stored in an XPDL file. Which are the process modeling classes (i.e. the classes used to represent the processes in memory)?
5	Which classes are initialized when the JSP container starts and are destroyed when it shuts down? These classes keep the long lived information and are used by almost all Web pages.

Table 4. Post-experiment survey questionnaire.

ID	Question
Q1	I had enough time to perform the lab tasks (1–5).
Q2	The objectives of the lab were perfectly clear to me (1–5).
Q3	The questions were clear to me (1–5).
Q4	I experienced no difficulty in reading the diagrams (1–5).
Q5	I experienced no difficulty in reading the source code (1–5).
Q6	How much time (as a percentage) did you spend looking at class diagrams? (A. <20%; B. >=20% and <40%; C. >=40% and <60%; D. >=60% and <80%; E. >=80%)
Q7	How much time (as a percentage) did you spend for source code browsing? (A. <20%; B. >=20% and <40%; C. >=40% and <60%; D. >=60% and <80%; E. >=80%)
Q8	I understood the meaning of Conallen's stereotypes (1–5).
Q9	I found Conallens stereotyped diagrams useful (1–5).
	1 = strongly agree, 2 = Agree, 3= Not certain, 4 = Disagree, 5 = strongly disagree.

Recall measures the fraction of expected items that are in the answer:

$$recall_{s,i} = \frac{|A_{s,i} \cap C_i|}{|C_i|}$$

Since the two above metrics measure two different concepts, it can be difficult to balance between them. We used an aggregate measure, $F-Measure$ [10], which is a standard combination of the two, defined as their harmonic mean:

$$F-Measure_{s,i} = \frac{2 \cdot precision_{s,i} \cdot recall_{s,i}}{precision_{s,i} + recall_{s,i}}$$

To obtain a single measure representing the comprehension level achieved by a student for an object application we use the mean of the F-Measure over all the questions. This study did not consider the time spent on answering questions as a dependent variable. Actually, all subjects worked for the full duration of the laboratories.

2.3 Experimental design and procedure

The assignment given to each group of subjects in each experimental session (*Lab 1* and *Lab 2*) follows the counterbalanced experimental design in Table 5. The design ensures that each subject worked on different *Systems* in the

Table 5. Experimental design.

	Group 1	Group 2	Group 3	Group 4
Lab 1	Claros-Con	Claros-UML	WfMS-Con	WfMS-UML
Lab 2	WfMS-UML	WfMS-Con	Claros-UML	Claros-Con

two *Labs*, receiving each time a different *Method* treatment. Also, the design permits us to consider different combinations of *System* and *Method* treatment in different order across *Labs*. More important, the chosen design permits the use of statistical tests (Two-Way and Three-Way ANOVA) for studying the effect of multiple factors. Subjects were split into four groups making sure that *high* and *low Ability* subjects were equally distributed across groups.

Before the experiments, subjects have been trained on Conallen's notation (also performing a simple understanding task on a small system related to an E-Shop), as well as all the technologies used in the target applications (e.g., Servlets/JSP). Also, the experiments were preceded by a detailed presentation of instructions related to the tasks to be performed, the goal of the experiment, but not the experimental hypotheses. Subjects were requested to work individually.

According to the design, each subject has been involved in two experimental sessions (laboratories), each lasting approximately 2 hours. Each laboratory consists of a com-
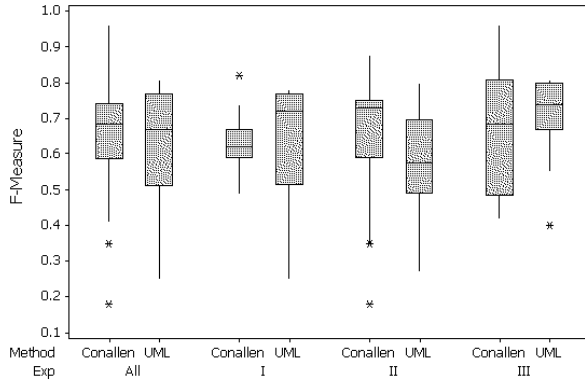
Figure 1. All data boxplots.

Table 6. F-Measure Descriptive statistics per Method, and p-values (one-tailed Mann-Whitney test).

Exp	UML			Conallen			p value	Std. eff. size
	Med	Mean	Std. Dev.	Med	Mean	Std. Dev.		
I	0.72	0.64	0.17	0.62	0.63	0.083	0.19	-0.03
II	0.57	0.58	0.15	0.73	0.67	0.16	**0.01**	0.54
III	0.74	0.71	0.12	0.68	0.67	0.16	0.25	-0.29
All	0.67	0.62	0.15	0.68	0.66	0.14	0.22	0.20

Table 7. Results of Wilcoxon test (one-tailed).

Exp	Subjects	Diff Median	Diff Mean	Diff Std. Dev.	p value	% of Positive Diff.	Std. eff. size
I	13	-0.11	-0.0057	0.21	0.61	38%	-0.03
II	20	0.057	0.084	0.18	**0.04**	60%	0.50
III	10	-0.085	-0.056	0.2	0.88	30%	-0.41
All	43	-0.04	0.024	0.2	0.26	47%	0.12

prehension task on the assigned *System* (Claros or WfMS) documented either by Conallen or pure UML diagrams. The comprehension task was carried out by answering 12 open questions (the same number of questions as in [19]) on the assigned *System*. Most of the questions refer to realistic program understanding scenarios. In fact, they were derived from change requirements of real Web applications published on SourceForge[3] (a big repository of open source projects). A sample of the questions for the WfMS application is shown in Table 3. To answer the questions, subjects had the possibility to look at the diagrams, to use the Web applications and to browse the source code.

After each laboratory session, subjects were asked to fill-in a survey questionnaire regarding the task and system complexity, the adequacy of the time allowed to complete the task and the usefulness of the provided diagrams. The questionnaire (shown in Table 4) consists of 7 common questions plus 2 questions (Q8 and Q9) answered only by subjects using Conallen diagrams. Answers to Q1-Q5 and to Q8, Q9 are on a Likert scale [13] from 1 (strongly agree) to 5 (strongly disagree). Answers to Q6, Q7 are based on a 5 points ordinal scale: $\{A, B, C, D, E\}$.

3. Experimental results

This section reports the results from all the three experiments, analyzing the effect on the dependent variable (*F-Measure*) of the main factor (*Method*) treatments and of other factors, mainly related to the subjects' ability and experience. Finally, results from the analysis of survey questionnaires are reported.

---

[3]http://sourceforge.net/

3.1 Influence of Method

We consider first hypothesis H_0. Figure 1 and Table 6 show boxplots, descriptive statistics and results of the unpaired, one-tailed Mann-Whitney test on all data from the three experiments. Results indicate only for Exp II a significant difference between Conallen and UML, rejecting the null hypothesis. Thus, while in Exp II, performed with *Undergraduate* students, Conallen had a significantly positive effect, this is not the case of the other experiments, where UML subjects performed even better (yet not significantly better) than Conallen subjects, with negligible/small effect sizes. We observe a negligible effect size for the first experiment, a positive (i.e., Conallen better than UML), medium sized effect for the second experiment and a negative, small effect for the third experiment. The effect size for the combined data is a positive, small one.

Since experiments were organized as longitudinal studies where each subject performed the task, over the two different *Systems*, with the two possible treatments for *Method* (Conallen and UML), it is possible to use a paired Wilcoxon, one-tailed test to analyze the differences exhibited by each subject for the two treatments. As shown in Table 7, also in this case, only for Exp II there is a significant difference between UML and Conallen (p-value=0.04). The median difference is positive for Exp II, while it is negative for Exp I and III. In fact, while in Exp I and III only 38% and 30% of the subjects considered for the paired analysis obtained a positive (yet sometimes small) difference, such a difference was higher for Exp II (60%). The proportion test (p-value=0.23) did not indicate equity in the proportion of positive differences across experiments. The effect size is negligible in the first experiment, medium in the second

Table 8. Results by Ability and Experience: descriptive statistics.

Experience	Ability	UML				Conallen			
		Observations	Median	Mean	Std. dev.	Observations	Median	Mean	Std. dev.
any	high	29	0.71	0.68	0.12	25	0.68	0.67	0.14
any	low	12	0.58	0.57	0.17	18	0.70	0.69	0.08
Graduate	any	28	0.73	0.67	0.15	27	0.63	0.65	0.13
Undergraduate	any	28	0.57	0.58	0.15	27	0.73	0.67	0.16
Graduate	high	19	0.74	0.71	0.1	17	0.61	0.64	0.15
Graduate	low	9	0.41	0.63	0.2	10	0.64	0.67	0.08
Undergraduate	high	10	0.64	0.62	0.16	8	0.73	0.72	0.11
Undergraduate	low	3	0.52	0.52	0.008	8	0.73	0.70	0.09

one and small in the third one. The effect size of the combined data is also negligible. The analysis of all data did not reveal any useful insights, this also because subjects' backgrounds and experiences are different, and because results from different experiments go towards opposite directions.

In summary, using either unpaired or paired statistical tests hypothesis H_0 could not be rejected in general; we could reject it only for Exp II, but not for the remaining two experiments.

3.2 Influence of Subjects' Ability and Experience

As shown in the previous subsection, the overall experimental data do not indicate any significant difference depending on the *Method* factor, except for Exp II. This suggests that, possibly, other factors could have influenced the results or interacted with the *Method*. In this section we consider the other detailed hypotheses H_{0e}, H_{0a}, and H_{0ea}. That is, we investigate the effect of factors related to subjects' *Experience* — *Undergraduate* students (*U*, Exp II) vs. *Graduate* students (*G*, Exp I and III) — and *Ability* (*high* or *low*). This analysis was performed on the whole data set, similarly to Wohlin *et al.* [22], who compared software development using C and C++. This permits the use of parametric statistics[4] and is possible since (i) the experiment design, material and procedure is exactly the same, (ii) the way subjects *Ability* has been evaluated is the same. The only substantial difference — which is anyhow considered as an experimental factor (i.e., the *Experience*) — can be found between Exp II subjects (*Undergraduate*) and Exp I and III subjects (*Graduate*).

Descriptive statistics for the F-Measure, classified according to *Ability* and *Experience*, are shown in Table 8. Table 9 shows the results of the two-way ANOVA by *Method & Experience* and by *Method & Ability*. Other than confirming the absence of a significant effect due by the *Method* factor, ANOVA indicates no direct, significant effect due to *Experience* or *Ability*. Nevertheless, in both cases there is a significant interaction of the latter factors with *Method*. At

least for *Experience*, this could have been expected since, as shown in the previous section, only Exp II subjects (*Graduate*) showed a significant difference between UML and Conallen.

Table 9. Two-way ANOVA.

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Method	1	0.025	0.026	1.22	0.27
Experience	1	0.046	0.046	2.20	0.14
Method:Experience	1	0.084	0.084	3.98	**0.049**
Residuals	106	2.24	0.021		

(a) Method & Experience

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Method	1	0.012	0.012	0.66	0.42
Ability	1	0.029	0.029	1.66	0.20
Method:Ability	1	0.085	0.085	4.86	**0.030**
Residuals	80	1.39	0.017		

(b) Method & Ability

The interaction can be better analyzed by looking at the interaction plots shown in Figure 2. The figure highlights a strong interaction between *Method* and *Experience* and between *Method* and *Ability*. Both figures consistently indicate that:

1. *high Ability* and *Graduate* subjects perform, on average, better using pure UML than using Conallen, while *low Ability* and *Undergraduate* subjects perform better using Conallen;

2. Conallen reduces the (mean) difference between *high* and *low Ability* subjects, as well as between *Graduate* and *Undergraduate*.

Three-way ANOVA by *Method, Experience* and *Ability* indicates no three-way interaction, while confirming the presence of two-way interactions. Therefore, we can reject the null hypotheses H_{0e} and H_{0a} while we could not find any statistical support to also reject H_{0ea}.
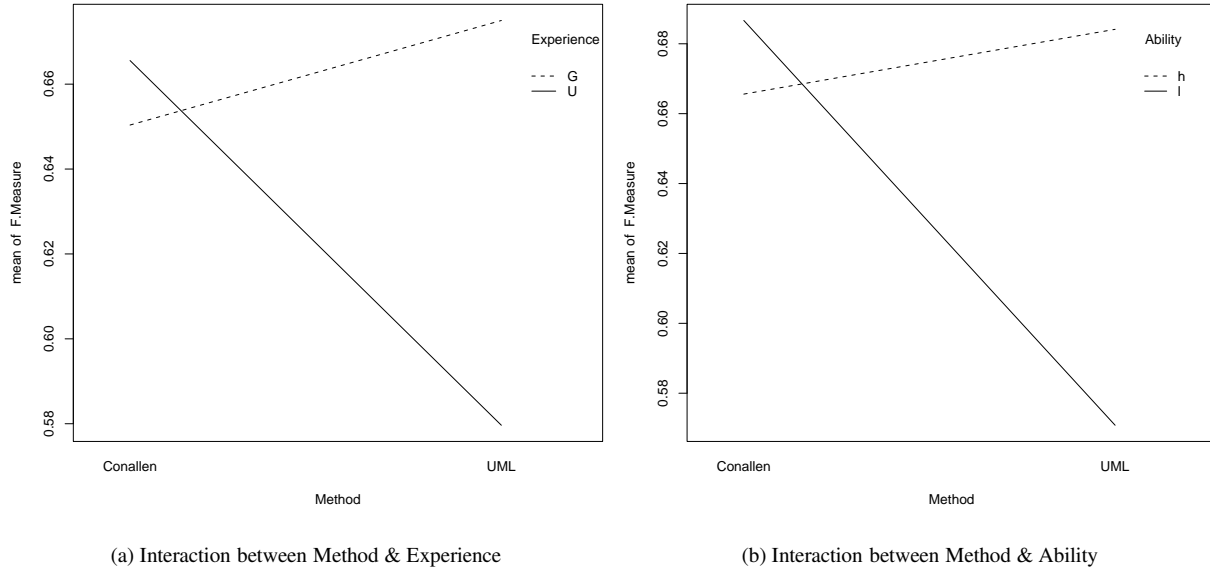
---

[4]Although to reduce the threats Mann-Whitney tests were also performed other than t-tests.

(a) Interaction between Method & Experience

(b) Interaction between Method & Ability

Figure 2. Interaction plots.

3.3 Effect of other factors

Other than *Experience* and *Ability*, other factors (*System, Lab, Experiment*) could have influenced the results. Nevertheless, ANOVA revealed no significant influence of such factors, nor any significant interaction with the *Method*.

3.4 Survey Questionnaire Results

To better understand the experimental results, we analyzed the feedback provided by subjects after each *Lab*. Statistical significance of differences has been tested using a Mann-Whitney one-tailed test. Detailed results can be found in a technical report [16].

By looking at the more general questions (**Q1–Q5**), we noticed that, overall, subjects had enough time to perform the task (**Q1**), even though, as expected, *Undergraduate* subjects had more difficulty (p-value=0.04) and the same happened for *low Ability* subjects (p-value=0.08). Also, both *Undergraduate* and *low Ability* subjects felt the task to be performed (**Q2**) significantly less clear than *Graduate* and *high Ability* subjects (p-value=6e-4 and 0.04 respectively). No major difference is found regarding the clarity of the comprehension question (**Q3**) nor for the difficulties in reading the source code (**Q5**).

Undergraduate subjects felt diagrams (**Q4**) more difficult to be understood than *Graduate* (p-value=0.0003), whilst at the end they were able to benefit of the use of diagrams, especially when diagrams made use of stereotypes.

It is also interesting to note how subjects (especially *Undergraduate*) felt diagrams less easy to understand than source code (p-value=0.03). Such a difference is even more evident for *Undergraduate* subjects, while there is no difference between *high* and *low Ability* subjects.

Questions related to the time spent by subjects on diagrams (**Q6**) and on source code (**Q7**) gave results that seem to be somewhat in contradiction with the quantitative results. *Graduate* subjects spent more time on diagrams when using Conallen than when using UML (p-value=0.0002), while this is not the case for *Undergraduate* subjects (p-value=0.5). Also, *Graduate* subjects spent more time on code than on diagrams when using UML (p-value=0.0003), while there is no significant difference for *Undergraduate* subjects (p-value=0.62). Despite that, quantitative results show that *Undergraduate* subjects benefited more of Conallen diagrams than *Graduate* subjects. While *high Ability* subjects did not spend significantly more time on diagrams than on code when using Conallen (p-value=0.15), this happened for *low Ability* subjects (p-value=0.01). When using UML, both subject categories spent more time on source code than on diagrams (p-value=0.003 for *high Ability* and 0.005 for *low Ability* subjects).

Differently from questions **Q6** and **Q7**, questions **Q8** and **Q9** confirm the quantitative results: By looking at **Q8** answers, it can be noted that *Undergraduate* subjects seem to have better understood the meaning of Conallen stereotypes than *Graduate* subjects (p-value=0.007). The same behavior, even if the difference is significant only at $\alpha = 0.1$,

is observed when comparing *low Ability* subjects with *high Ability* subjects (p-value=0.1). Finally, **Q9** indicates that *Undergraduate* subjects and *low Ability* subjects felt stereotypes more useful than *Graduate* (p-value=0.005) and *high Ability* subjects (p-value=0.03) respectively.

4. Discussion

Results showed no significant effect on the mean subjects' performance dependent on the *Method* factor. It is not possible to conclude that the use of stereotypes significantly improves their performance. When analyzing the effect of subjects' *Experience* and *Ability*, we found a significant interaction between each of the two factors and the *Method*. Interaction plots consistently indicated that *low Ability/Experience* subjects significantly benefited from stereotypes, while *high Ability/Experience* subjects' performance was substantially the same with and without relying on stereotypes (slightly better without, but no statistical significance was found). *low Ability/Experience* subjects felt the task more difficult to perform and, although they had some difficulty in using the diagrams, they were able to rely on stereotyped diagrams to gain a benefit and, in the end, felt these diagrams more useful than *high Ability/Experience* subjects. The latter had less difficulties in performing the task and, whilst they spent more time on diagrams when using stereotypes, they did not really felt the diagrams useful for answering the questions: they relied on the source code.

By looking at the use subjects made of diagrams (according to what reported in survey questionnaires), the following conclusion can be drawn: subjects tend to spend more time on diagrams if these are stereotyped; nevertheless this does not necessarily lead to a significant improvement of the software comprehension. The improvement is only observable for *low Ability/Experience* subjects.

Results also showed that the use of stereotypes consistently reduced the gap between *high* and *low Ability/Experience* subjects. This result can have an important, practical effect: whenever an organization cannot afford high skilled developers, the use of stereotypes may contribute to making low ability maintainers more effective. This is possible because, once developers know basic UML, learning the use of stereotyped notations is pretty straightforward, as demonstrated by the fact that even *low Ability*, *Undergraduate* subjects were able to quickly become productive with them.

In summary, the following observations can be derived from the results:

1. Experts use the code more extensively and diagrams do not make a significant difference for them.

2. Stereotypes help and support substantially novice programmers; and

3. For the above two reasons, stereotypes reduce the performance gap between novices and experts.

Our interpretation of the latter fact is that experts tend to adopt a more "integrated" approach (compared to the top-down approach naturally supported by stereotyped diagrams). The capability to quickly browse code never seen before requires experience, and varies a lot depending on how capable a maintainer is. Vice versa, locating concepts into stereotyped diagrams is (relatively) simpler. Regardless of the *Ability* difference, this makes the knowledge level almost similar for everybody. When stereotypes are not available, the source code is the only place where some information related to the View (e.g., related to page links, Web form fields, HTTP sessions) can be located, and *low Ability/Experience* subjects result were penalized. Instead, when stereotypes are available, *low Ability/Experience* subjects successfully use them, making the difference with *high Ability/Experience* subjects not significant.

5. Threats to validity

This section discusses the threats to validity that can affect our results: *conclusion*, *construct*, *internal* and *external* validity threats.

Conclusion validity concerns the relationship between the treatment and the outcome. Attention was paid to not violate assumptions made by statistical tests. Whenever conditions necessary to use parametric statistics did not hold (e.g., analysis of each experiment data), we used non-parametric tests, in particular Mann-Whitney test for unpaired analyses and Wilcoxon test for paired analyses. Unless differently indicated, results were intended as statistically significant at $\alpha = 0.05$. The measure chosen to evaluate the comprehension, i.e., the F-Measure, allowed to evaluate the questionnaire answers in an objective manner, avoiding to give subjective scores. The comprehension questionnaire covered different aspects of the system, so that the high number of correct answers indicates a good comprehension level. Survey questionnaires, mainly intended to get qualitative insights, were designed using standard ways and scales [13]. This allowed us to use statistical test for also analyzing differences in the feedbacks. Finally, we dealt with random heterogeneity of subjects by introducing the *Ability* and *Experience* factors, and analyzing their interaction with *Method*, as well as the three-way interaction among the three factors.

Construct validity threats concern the relationship between theory and observation. Being an aggregate measure of precision and recall, the F-Measure well reflects the completeness and accuracy of questionnaire answers that,

on their own, provide an indication of the comprehension level achieved. Interactions between different treatments were mitigated by the chosen experimental design. Regarding the levels of the *Ability* factor, more levels than *high* and *low* could have been used. Nevertheless, analysis performed with more levels did not yield any different or contrasting result. We are aware, however, that *Ability* measures based on factors different than exam grades could have lead to different conclusions. To avoid social threats due to evaluation apprehension, students were not evaluated on their performance. Finally, subjects were not aware of the experimental hypotheses.

Internal validity concerns external factors that may affect an independent variable. A major threat is related to a number of students that, in Exp II and III, did not participate to both *Labs*. Paired analyses were limited to students that participated to both two *Labs*, while unpaired tests were used over all data available, including students who participated to one *Lab* only. These subjects were excluded from the analysis of the *Lab* factor. Other *internal validity* threats can be due to the learning effect experienced by subjects between *Labs*. Such an effect is mitigated by the chosen experimental design: subjects worked, over the two *Labs*, on different systems with different levels of the main factor (UML vs. Conallen). Also, there is the risk that, during labs, subjects might have learned how to comprehend the source code of a Java Web application and how to read UML diagrams. We limited this effect by means of a preliminary training phase. Finally, ANOVA analyses were used to study the influence of the *Lab* factor, for which no significant effect was found.

External validity concerns the generalization of the findings. This kind of threat is always present when experimenting with students. The selected subjects represent a population of students specifically trained on Web development technologies and software engineering methods. Also, all students involved in Exp I and III (graduate students) either had some professional experiences or worked on industrial project during their thesis. This makes these students comparable to industry junior developers. Nevertheless, the working pressure and the overall environment in industry is different, thus replicating the study in industry is highly desirable. The experiment objects were two real, though small, Web applications belonging to different domains. This makes the context quite realistic, despite further studies with different types of systems and with stereotypes related to different domains are necessary to confirm or confute the obtained results.

6 Related Work

The usefulness of graphical elements in software architecture diagrams has been assessed experimentally by Bratthall and Wohlin [2], who compared ten different representations aiming at enriching the design with qualitative information. We share with them the conclusion that graphical information provides an important support during software understanding. Staron *et al.* [19] conducted a study, for some aspects similar to ours, aiming at experimenting the usefulness of UML stereotypes related to the telecommunication domain. The study consisted of a set of experiments in academic and industrial environments. They found that stereotypes significantly helped both students and industrial programmers. Whilst we share with them the conclusion about the usefulness of stereotypes, our study presents two main differences: (i) our experimentation discusses the effect of ability and experience on the usefulness of stereotypes; and (ii) subjects, as happens in a realistic maintenance task, had also the source code available, other than UML diagrams. Our conclusions regarding undergraduate students are consistent with those of Staron *et al.* for subjects having a similar background. In our case, however, high ability and experienced subjects did not benefit of stereotypes: we believe this was due to the different treatment, i.e., more complex diagrams and the availability of source code, that high ability and experienced subjects were able to exploit as effectively as diagrams. In a companion paper [15] we presented the results of the second experiment alone, without however discussing the effect of subjects' ability and experience.

Experiments aiming at studying the impact of UML documentation in software maintenance [1] indicate that such a documentation improves the functional correctness of changes and the quality of the design. While simple class diagrams, with or without stereotypes, help low ability or low experience subjects, a complete, thorough UML documentation requires a certain learning curve to become useful [1]. In fact, in some cases the previous experience of subjects influences the understandability of UML diagrams. Torchiano [21] showed that object diagrams have a significant impact on comprehension tasks, when compared with UML documentation consisting of class diagrams only.

UML limitations in aiding program understanding are highlighted in experiments performed by Tilley and Huang [20]. They highlight that UML does not provide a sufficient support to represent domain knowledge, support that stereotypes surely contribute to improve. Lemus *et al.* [8] showed that composite states improve the understandability of statecharts provided that subjects had a previous experience in using them. This also happens when UML is complemented with complex formalisms, such as the Object Constraint Language (OCL) [3]: a substantial training is required to make OCL useful, although for some tasks, such as defect detection, an interaction between ability and treatment similar to what obtained in our study was detected. OCL better helped low ability subjects, who were

not able to guess system functionality from the textual description.

As it happened in our study, the most intuitive notation is not always the one producing better performances. Purchase *et al.* [14] compared different UML syntaxes for class relationships, and found that the less intuitive ones helped more the subjects, since this forced them to be more diligent in the analysis. Finally, consistently with our results, Lawrance *et al.* found that graphical visualizations of code coverage information helps end-user programmers, while the same does not happen for professional developers [12].

The aforementioned studies do not deal with the influence of the subjects' ability and experience on the comprehension level. The impact of subjects' background on pair design was investigated by Canfora *et al.* [5]. Authors found that, when building pairs, it should be avoided to put together subjects having a different background.

7. Conclusions and future work

This paper 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).

Results showed that stereotypes improve significantly the performance of low ability/low experience subjects. Correspondingly, companies might consider them useful for knowledge transfer from experts to novice developers. Work-in-progress is devoted to replicating the study with professionals, as well as to investigating whether the use of stereotypes affects the execution of maintenance (instead of comprehension) tasks, in terms of effort and quality of the maintained artifacts.

References

[1] E. Arisholm, L. C. Briand, S. E. Hove, and Y. Labiche. The impact of UML documentation on software maintenance: An experimental evaluation. *IEEE Transactions on Software Engineering*, 32(6):365–381, 2006.

[2] L. Bratthall and C. Wohlin. Is it possible to decorate graphical software design and architecture models with qualitative information? -An experiment. *IEEE Trans. Softw. Eng.*, 28(12):1181–1193, 2002.

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

[4] D. Brugali and M. Torchiano. *Software Development: Case Studies in Java*. Addison Wesley, 2005.

[5] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio. Confirming the influence of educational background in pair-design knowledge through experiments. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1478–1484. ACM Press, 2005.

[6] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002.

[7] J. Conallen. *Building Web Applications with UML, Second Edition*. Addison-Wesley Publishing Company, Reading, MA, 2002.

[8] J. A. Cruz-Lemus, M. Genero, M. E. Manso, and M. Piattini. Evaluating the effect of composite states on the understandability of UML statechart diagrams. In *proceedings of the International Conference on Model Driven Engineering Languages and Systems (MODELS 2005)*. Springer, 2005.

[9] O. M. F. De Troyer and C. J. Leune. WSDM: a user centered design method for web sites. In *Proceedings of the seventh international conference on World Wide Web 7*, pages 85–94. ACM Press, 1998.

[10] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.

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

[12] J. Lawrance, S. Clarke, M. M. Burnett, and G. Rothermel. How well do professional developers test with code coverage visualizations? an empirical study. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2005), 21-24 September 2005, Dallas, TX, USA*, pages 53–60, 2005.

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

[14] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, and C. Britton. UML class diagram syntax: an empirical study of comprehension. In *APVis '01: Proceedings of the 2001 Asia-Pacifi c symposium on Information visualisation*, pages 113–120, Darlinghurst, Australia, 2001. Australian Computer Society, Inc.

[15] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. An empirical study on the usefulness of Conallen's stereotypes in Web application comprehension. In *Proceedings of the international Symposium on Web Site Evolution (WSE 2006)*, pages 58–65. IEEE Computer Society, 2006.

[16] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. The role of experience and ability in comprehension tasks supported by UML stereotypes. Technical report, RCOST - University of Sannio, Italy, Sep 2006 *http://www.rcost.unisannio.it/mdipenta/icse07tr.pdf*, 2006.

[17] J. Rumbaugh, I. Jacobson, and G. Booch. *Unifi ed Modeling Language Reference Manual*. Addison-Wesley, 2004.

[18] D. Schwabe and G. Rossi. An object oriented approach to web-based application design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.

[19] M. Staron, L. Kuzniarz, and C. Wohlin. Empirical assessment of using stereotypes to improve comprehension of uml models: a set of experiments. *Journal of Systems and Software*, 79:727–742, 2006.

[20] S. Tilley and S. Huang. A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding. In *SIGDOC '03: Proceedings of the 21st annual international conference on Documentation*, pages 184–191, New York, NY, USA, 2003. ACM Press.

[21] M. Torchiano. Empirical assessment of UML static object diagrams. In *Proceedings of the International Workshop on Program Comprehension*, pages 226–229. IEEE Computer Society, 2004.

[22] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.