An empirical study on the usefulness of Conallen's stereotypes in Web application comprehension

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

Comprehension of Web applications is a complex task, since several concerns co-exist in their implementation, among which the business logic, the navigation structure (as supported by hyperlinks and form submission), and persistent data storage. Design notations tailored for Web applications promise increased understandability and maintainability, thanks to the explicit representation of Web specific elements (such as hyperlinks and forms).

In this paper, we report the results obtained from the execution of an empirical study involving comprehension tasks on two Web applications. Assuming the availability of the source code, two forms of design diagrams have been recovered from the code: standard UML diagrams and UML diagrams extended with Conallen's stereotypes. The research question addressed by this study is whether enriching standard UML diagrams with Web specific stereotypes gives any significant contribution to the understandability of the Web applications.

Keywords: Empirical Studies, Web Applications, Design Notations.

1 Introduction

The complexity of Web applications and the difficulties encountered during their evolution descend from the multiple, different concerns that are handled in the implementation, where they are tangled with each other. Any realistic Web application includes persistent data, business logic, navigation structure and user interface. Other relevant concerns include security and authentication.

During the initial development of Web applications, design notations and methodologies can be used to model the different concerns separately. Among the most referenced approaches are WebML [2], WSDM [12], OOHDM [10], Conallen [3]. Many of these notations are extensions of UML [8].

In practical cases, design models may be unavailable or inconsistent with the code when the Web application has evolved for some time. In such a context, it is still possible to take advantage of the separation of concerns supported by design models, by recovering them semi-automatically from the code through reverse engineering. However, both keeping the design models aligned with the implementation and reverse engineering them from the implementation have a non negligible cost, which might be justified only by increased ease of comprehension and evolution.

In this paper, we offer some insights on the actual benefits of reverse engineered design diagrams in the execution of comprehension tasks. For this purpose, we instantiate (see Table 1) the generic experimental framework, which was proposed by Tonella *et al.* [11] with the aim of supporting the empirical validation of design notations for Web applications. The goal of the experiment reported in this paper was to evaluate the effectiveness of Conallen's stereotypes (WAE – Web Application Extension – notation [3]) in improving the comprehension of Web applications. The WAE notation was compared to the basic (standard) UML notation, by conducting an experiment with Bachelor students who were required to understand some features of two Web applications, simulating a typical Wed evolution scenario, where comprehension is the first step before change implementation. Results indicate that Conallen's diagrams provide a (statistically) significant support to Web application understanding, which may justify the reverse engineering effort.

The paper is organized as follows: Section 2 discusses related works and gives an overview of the two compared notations. Section 3 describes the design of the empirical study that we conducted. Results are presented in Section 4. Conclusions and future works are given in Section 5.

Goal	Analyze the support given by Conallen's stereotypes [3] to the comprehension of Web applications.
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 vs. Conallen's stereotypes.
Other factors	Systems, subjects and subject skills.
Dependent variables	Comprehension level.

Table 1. Overview of the experiment.

2 Background

2.1 Related work

A similar, preliminary study, focused on the use of stereotypes for comprehending Object-Oriented applications in the telecommunication domain has been conducted by Kuzniarz *et al.* [6]. The authors showed that the use of stereotypes help to improve the comprehension. The main differences from the present study are:

1. programming style (OO vs. Servlet/JSP Web application);
2. size and complexity of the considered applications (the applications in [6] count 14 classes, while the two systems we used have 78 and 92 classes and JSPs);
3. application domain;
4. stereotypes being evaluated (ad-hoc stereotypes introduced by Kuzniarz *et al.* [6] for the telecommunication domain vs. Conallen's stereotypes [3]). While the mapping between a basic UML model and a stereotyped model is one–to–one for the telecommunication stereotypes, this is not the case for Conallen's diagrams, that also model as UML classes artifacts (e.g., client pages, forms, scripts) not appearing in basic UML models, and contains some associations (e.g. hyperlinks) not visible in basic UML models; and
5. above all, while in the Kuzniarz *et al.* experiment subjects just relied on diagrams, in our experiment they also had the source code available. This is in our opinion more realistic for a software maintenance task.

2.2 Web Design Notations

Among the Web design notations proposed in the literature [2, 3, 10, 12], in this paper we focus on WAE [3], the notation proposed by Jim Conallen and consisting of a set of stereotypes that extend UML so as to support the graphical representation of the navigation structure and of the dynamic page construction relationships of a Web application. We chose this notation because of its popularity and because it extends a well-known and widely used design notation, UML [8]. In our future work we intend to consider more design notations in our evaluations.

Figure 1 gives an example of the extra information provided by Conallen's diagrams, compared to that usually represented in standard UML class diagrams. The modeled Web application implements a glossary. On the left is the basic UML diagram, showing the Servlet (*GetEntries*) and the database. On the right, the same diagram is enriched with Conallen's notation. It includes the client pages generated by the Servlets (e.g., *EntryListing*), the static pages (*Glossary home*) and the hyperlinks (notation: `<<link>>`).

3 Experiment definition, design and settings

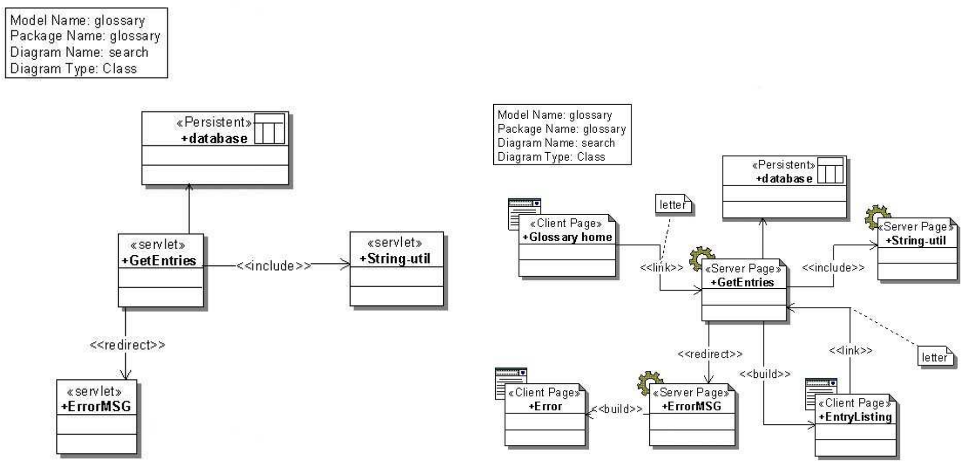
In this section, we describe in detail the definition, design and settings of the proposed experiment, following the guidelines by Wohlin *et al.* [13] and Juristo and Moreno [5] on how to document and report empirical studies in software engineering. Table 1 summarizes the main elements of the experiment.

The *goal* of the study is to analyze the use of stereotyped UML diagrams (following the approach by Conallen [3]), with the purpose of evaluating their usefulness in Web application comprehension. The *context* is one where the source code is the main repository of information about the system. Design diagrams can are recovered semi-automatically from the code by means of reverse engineering. The *quality focus* is ensuring high comprehensibility and maintainability, while the *perspective* is both of *Researchers*, evaluating how effective are the stereotyped reverse engineered diagrams during maintenance, and of *Project managers*, evaluating the possibility of adopting a Web application design and reverse engineering tool in her/his organization.

3.1 Hypotheses

We are mainly interested in how stereotypes affect comprehension. Thus we formulate the null hypothesis that no difference is observed using either notation. When the null hypothesis can be rejected with relatively high confidence (we set $\alpha = 5\ \%$ in our case), it is possible to formulate an alternative hypothesis, which typically admits a positive effect of one design notation in the execution of understanding tasks. The detailed hypotheses are:

H_0 When doing a comprehension task the use of stereotyped reverse engineered class diagrams (versus non-stereotyped reverse engineered class diagrams) **does not significantly affect** the comprehension level.



(a) UML (b) Conallen

Figure 1. Basic UML class diagram (left) compared to Conallen's diagram (right).

H_a When doing a comprehension task the use of stereotyped reverse engineered class diagrams (versus non-stereotyped reverse engineered class diagrams) **significantly affects** the comprehension level.

3.2 Treatments

The treatment considered in this experiment is WAE, the design notation proposed by Conallen [3]. Since this notation extends UML through a set of stereotypes, the notation used for comparison (second treatment) is basic UML, with no Web-specific stereotype. The aim is to determine whether a significant improvement can be obtained by means of Conallen's stereotypes in the maintenance and evolution phase.

Diagrams are reverse engineered automatically from the code and then adjusted manually, 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.

3.3 Objects

Two Web applications were selected for this study: *Claros*[1] and *WfMS*[2] [1]. Both are small/medium size applications (see Table 2) based on the Servlet/JSP technology and downloaded from the Internet. Although commercial or institutional Web applications may be larger, given the time constraints of the experiment and the involved subjects (students), it was not feasible to consider larger examples. The application domains of the selected system 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 View (and thus more complex navigational model). The View comprises 19 UML classes (38 in the Conallen's diagram) for Claros and 13 (24 in the Conallen's diagram) for WfMS.

[1]http://www.claros.org

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

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

Table 2. Characteristics of the systems under study.

3.4 Subjects

The subjects participating in the study are the 35 Bachelor students attending the Laboratory of Software Engineering course (2nd year B.Sc.) at the University of Trento, Italy. All the students registered for this course are from the same class, with more or less the same background. Partic-

	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

Table 3. Experimental design.

ipants had attended previously Programming and Software Engineering courses. They had a good knowledge of UML and Java.

3.5 Procedure and design

Students have been trained on Conallen's notation, as well as all the technologies used in the target applications (e.g., Servlets/JSP). They have been involved in two experimental sessions (laboratories), each lasting approximately 2 hours. The assignment given to each group of students in each laboratory follows the experimental design in Table 3, which is an instance of the counter-balanced scheme described in [11]. Students were assigned randomly to the four groups[3].

Each laboratory consists of a comprehension task on the assigned object (Claros or WfMS) documented either by Conallen (Con) or pure UML (UML) diagrams. The comprehension task is carried out by answering 12 open questions (the same number of questions as in [6]) on the assigned system. Most of the questions are realistic scenarios in a program understanding task. To formulate them we have referred to some change requirements of real web applications contained in SourceFourge[4] (a big repository of open source projects). In order to answer the questions, students had the possibility to look at the diagrams, to use the Web applications, and to browse the source code. Each object (Web application) is associated with a specific set of questions. A sample of the questions for the WfMS application is shown in Table 4.

After each lab session, we asked the students 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 5) consists of 7 common questions plus 2 questions (Q8 and Q9) answered only by students using Conallen diagrams. Answers to Q1-Q5 and to Q8, Q9 are on a Likert scale [7] 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]Students have to work individually. The number of students per group ranges between 6-8.

[4]http://sourceforge.net/

3.6 Variables

In this study we have only one factor, the *Method*, which indicates the notation used to describe the web application. It can assume one of the values in $\{$*UML, Conallen*$\}$. The main outcome observed in the study is *comprehension level* that will be measured by three variables. To measure the comprehension level, we assessed the answers to the questionnaires using an information retrieval approach. Since the answer to each question has to be expressed as 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 [4]. Precision measures the fraction of items in the answer that are correct:

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

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 added a derived measure, $F{-}measure$ [4], which is a standard metrics defined as the harmonic mean of $precision$ and $recall$:

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

To obtain a single number representing the comprehension level achieved by a student for an object application we use the mean of the F-measure over all the questions.

4 Experimental Results

This section summarizes the main results obtained from the experimentation we performed. Some insights can be obtained by looking at the descriptive statistics in Table 6 and at the boxplots in Figure 2, that compare the F-measure the precision and the recall between subjects using basic UML and subjects using Conallen's stereotypes. The figures clearly highlight the benefits obtained when Conallen's stereotypes are used. Unpaired (Mann-Whitney) statistical tests supports such an evidence, with p–value=0.01 for the

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

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.

Table 5. Post-experiment questionnaire.

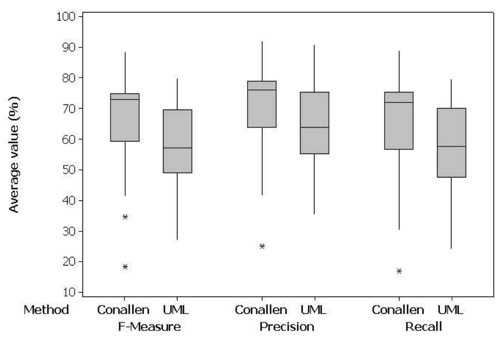


Figure 2. Boxplot of F-Measure, Precision and Recall.

Statistics	Method	Median	Mean	Std. Dev
F-Measure	Conallen	73.08	66.62	15.51
	UML	57.18	57.91	14.87
Precision	Conallen	76.04	70.54	14.65
	UML	63.89	63.99	15.26
Recall	Conallen	71.92	66.72	17.21
	UML	57.71	57.73	15.20

Table 6. Descriptive statistics.

Variable	**Median**	**Mean**	**Std Dev.**	**p–value**
F-Measure	5.65	8.38	18.36	**0.045**
Precision	9.68	6.72	19.06	0.066
Recall	5.03	8.35	19.11	0.053

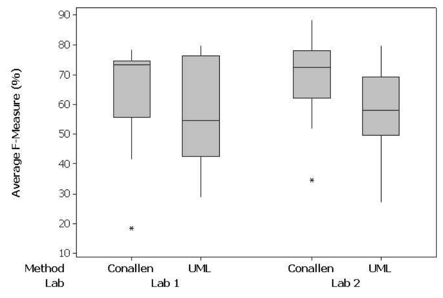
Table 7. Wilcoxon paired test.

F–Measure, 0.036 for the Precision and 0.009 for the Recall. Unpaired tests were used to gain statistical evidence over the whole set of subjects involved in the experiments. In fact, 27 subjects out of 35 who signed participated to *Lab* 1 and 29 participated to *Lab 2*. 20 subjects participated to both laboratories. Note that we used a one–tailed test being only interested in the difference in one direction, i.e., whether or not the Conallen's notation introduces benefits.

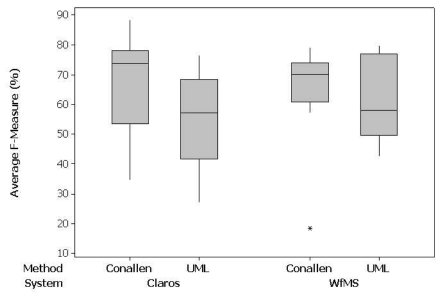
The chosen experiment design also allowed us to perform paired statistical tests (the Wilcoxon paired test)[5], comparing the performance of each subject with the two treatments (Conallen and UML). As shown in Table 7, the use of Conallen stereotypes introduces a significant improvement over the F–Measure, and a marginal significance for Precision and Recall separately.

Overall, the statistical tests performed allow us to reject the null hypothesis stated in Section 3. In other words, results from this experiment show a significant improvement of the comprehension level when using diagrams with

[5]Over the 20 subjects present at both laboratories.



(a) Lab



(b) System

Figure 3. Boxplots of F–Measure per Lab (a) and System (b).

Conallen's stereotypes.

In the following, we will analyze the effect of other factors, namely *System* and *Lab*, over the obtained results. Due to space limitations, we report analyses related to the F–Measure only. Also, analysis of other factors (e.g., *Skills* or *Time* necessary to complete the task) cannot be included for the same reason.

Table 8 reports results from two-way Analysis Of Variance (ANOVA) by *Method and System* and by *Method and Lab*. Both analyses indicate no significant influence of *System* and *Lab* over the main factor (*Method*) that, instead, is significant. Also, there is no significant interaction of the *Method* with the other two factors.

Variable	Df	Sum Sq	Mean Sq	F value	p–value
Method	1	1043.0	1043.0	4.4634	**0.03954**
System	1	29.2	29.2	0.1249	0.72528
Method:System	1	283.7	283.7	1.2142	0.27567
Residuals	51	11917.8	233.7		

(a) F-Measure by Method and System

Variable	Df	Sum Sq	Mean Sq	F value	p–value
Method	1	1043.0	1043.0	4.3902	**0.04113**
Lab	1	74.5	74.5	0.3137	0.57790
Method:Lab	1	39.7	39.7	0.1669	0.68458
Residuals	51	12116.5	237.6		

(b) F-Measure by Method and Lab

Table 8. Two-way ANOVA.

Figure 3 shows the effects of the *Method* in the two *Labs* (a) and for the two *Systems* (b) using boxplots. The boxplots highlight how the difference between Conallen and UML is more evident – and, above all, significant – in *Lab 2* (p–value 0.016) than in *Lab 1* (p–value 0.20). Also, while the

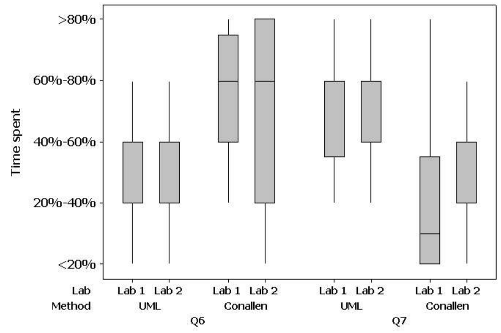


Figure 4. Time spent on diagrams (Q6) and source code (Q7).

difference for WfMS is not significant (p–value 0.15) such difference is significant for Claros (p–value 0.01). Further analyses showed that, in the case of WfMS, the difference is not significant in *Lab 1*, while it is marginally significant in *Lab 2* (p–value 0.064). Note that, since here four different tests were performed to make an analysis by *Lab* and by *System*. The significant p-values showed here are significant according to the Bonferroni correction:

$$\alpha_{Bonferroni} = \frac{5\%}{4} = 1.25\%. \tag{1}$$

4.1 Analysis of Survey Questionnaires

The analysis of the survey questionnaires that the subjects filled-in after each experiment can be useful to better understand the experimental results. Analyses are supported by descriptive statistics and results of Mann–Whitney test, for which the p–value is reported.

Overall, all subjects agreed they had enough time to perform the task (**Q1**, median=2), especially in *Lab* 2 (p–value 0.05). The objectives were clear enough (**Q2**, median=2), and became marginally clearer from *Lab* 1 to *Lab* 2 (p–value 0.07). Similar agreement levels are obtained for the clarity of the questions (**Q3**, median=2) and, also in this case, there was a significant improvement between *Lab* 1 and *Lab* 2 (p–value 0.05).

Subjects felt an average difficulty (median=3 in both cases) when understanding diagrams (**Q4**) and understanding code (**Q5**). No significant improvement was visible across *Labs*, while it is possible to note a marginal difference between the code of Claros and WfMS (the latter appears to be more complex, p–value 0.07). Finally, confirming the usefulness of Conallen's diagram, subjects had less difficulty to understand the UML diagrams when the Conallen's stereotypes were present (p–value=0.03).

Subjects spent between the 40% and 60% of their time by looking at diagrams (**Q6** and **Q7**). Such a percentage did not change across *Labs*, whilst subjects working on WfMS spent a bit more time over diagrams. What, once more confirms the quantitative results, is that subjects using Conallen's stereotypes spent significantly more time (between 60% and 80%) on diagrams than those using basic UML (p–value $< 4 \cdot 10^{-5}$) and, consequently, significantly less time on the source code (see also boxplots in Figure 4).

Finally, subjects using Conallen's stereotypes agreed they understood the notation well (**Q8**, median=2) and found the stereotypes useful to support the task they performed (**Q9**, median=2). No significant difference can be observed across *Labs* and *Systems*.

4.2 Discussion

The main result of this study is that Conallen's diagrams provide a significant support to Web application understanding. We can interpret this result in terms of the Web application features that are represented explicitly in Conallen's diagrams. The navigation structure, page generation and form submission are immediately apparent from Conallen's diagrams, while they are absent in basic UML diagrams. As a consequence, the subjects using basic UML were forced to go to the code in order to gather the same information. This is confirmed by the answers to questions Q6 and Q7 of the survey: subjects using basic UML spent significantly more time on the code than subjects using Conallen. Thus, Conallen's explicit representation of Web-specific notions gave a relevant contribution to the comprehension of the considered Web applications.

The difference between Conallen and basic UML is more evident in *Lab* 2 than in *Lab* 1 and for Claros than for WfMS. The observed difference between the two laboratories can be explained by a learning effect. When performing the task of the second laboratory, subjects seem to have acquired some knowledge on how diagrams can be useful and can effectively support the completion of the assigned task. They become more familiar with the experimental setting and with the activities to be carried out. Consequently, they exploit the information available from the diagrams at best. This is confirmed by some answers to the survey questionnaire (e.g., Q1, Q2, Q3), where adequacy of the allowed time, clarity of objectives and clarity of questions increased from *Lab* 1 to *Lab* 2.

The difference between Claros and WfMS can be explained by the specific features of these two applications. Claros is larger and more complex, thus diagrams are expected to be more beneficial for it. Moreover, the navigation structure prevails over the business logic for Claros, while the opposite is true for WfMS, so that Conallen's diagrams are expected to be more informative for Claros. Finally, subjects were more familiar with the application domain of Claros (Web mail) than WfMS (workflow management), so that they had to spend more time on the code to understand the business logic with WfMS (see also the answers to Q6 and Q7).

4.3 Threats to Validity

This Section discusses the threats to validity that can affect our results: *internal*, *construct*, *conclusion* and *external* validity threats [13].

Internal validity threats can be due to the learning effect experienced by subjects between *Labs*. This is mitigated thanks to the experiment design: subjects worked, over the two *Labs*, on different systems and using two different levels of the main factor (UML vs. Conallen). Nevertheless, there is still 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 tried to limit this effect by means of a preliminary training phase. Subjects were previously trained on the relevant topics. Moreover, the *Lab* factor has been accounted as a factor in the analysis of results. ANOVA showed no significant effect due to *Lab*, although the differences between UML and Conallen were more evident in Lab 2, as discussed above.

Construct validity threats that may be present in this experiment, i.e., interactions between different treatments, were mitigated by a proper design that allowed to separate the analysis of the different factors and of their interactions. To avoid social threats due to evaluation apprehension, students were not evaluated on their performance in the *Lab*. Finally, subject were not aware of the experimental hypothesis.

About *conclusion validity*, proper tests were performed to statistically reject the null hypothesis. In particular the chosen experiment design permitted the use of paired tests, although results of unpaired tests were obtained over a larger data set (see Section 4). In cases where differences were present but not significant, this was explicitly mentioned. Non–parametric tests were used in place of parametric tests where the conditions necessary to use parametric tests do not hold. Also, ANOVA results were confirmed by non-parametric tests (Friedman test). The measure chosen to evaluate the comprehension, i.e., Precision, Recall and 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 [7]. This allowed us to use statistical test to analyze also differences in the feedbacks.

Last, but not least, *external validity* threats are always present when experimenting with students. Since the selected subjects represent a population of students specifically trained on Web development technologies and software engineering methods we expect a similar trend of improvement for industrial developers [9]; although only further specific studies can confirm or contradict the obtained results. The experiment objects were two real Web applications belonging to different domains. This makes the context quite realistic, despite only further studies with different types of systems can confirm the obtained results.

5 Conclusions and work–in–progress

Web applications are complex software systems which involve at least one additional dimension over traditional systems, navigation through hyperlinks and form submission. Understanding these applications is a challenging task and standard design notations such as basic UML provide little support. Experimental data indicate that diagrams with Web-specific stereotypes, such as Conallen's, provide a substantial, significant contribution to the comprehension activity. Reverse engineering meaningful Conallen diagrams from the code requires some effort, devoted to the selection of the appropriate views and of the information to show or hide. However, our results indicate that such an effort has a statistically significant, positive effect on the successive understanding activities.

As it always happens with empirical studies, replication in different contexts, with different subjects and objects, is the only way to corroborate our findings. It would be interesting to consider alternative experimental settings in several respects, but maybe the most important one is the profile of the involved subjects. We considered students of the 2nd year of the B.Sc. degree. Replicating this study with students of the master degree, with graduated students and with professionals would be extremely important to understand how these different sub-populations of programmers make use of the information provided by Conallen's diagrams. Novices are expected to behave quite differently from expert programmers. This kind of studies is part of the agenda of our future work.

Acknowledgments

We are grateful to Anna Perini and Angelo Susi, the teachers of Laboratory of Software Engineering, who accepted to host the experiment inside their course. Luana Stedile contributed to the in field execution of the experiment and to the collection of the data. Finally, we thank all the students who participated in the experiment. Without them this work would not have been possible.

References

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

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

[3] J. Conallen. *Building Web Applications with UML*. Addison-Wesley Publishing Company, Reading, MA, 2000.

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

[5] N. Juristo and A. Moreno. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, Englewood Cliffs, NJ, 2001.

[6] L. Kuzniarz, M. Staron, and C. Wohlin. An empirical study on using stereotypes to improve understanding of uml models. In *Proceedings of the International Workshop on Program Comprehension (IWPC)*, pages 14–23, Bari, 2004.

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

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

[9] P. Runeson. Using students as experiment subjects - an analysis on graduate and freshmen student data. In *Intl Conf. Empirical Assessment and Evaluation in Software Eng. (EASE03)*, pages 95–102, 2003.

[10] 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.

[11] P. Tonella, F. Ricca, M. Di Penta, and M. Torchiano. Towards empirical validation of design notations for web applications: An experimental framework. In *Int. Workshop on Web Maintenance and Reengineering*, March 2006.

[12] O. M. F. D. 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.

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