Towards Empirical Validation of Design Notations for Web Applications: An Experimental Framework

Paolo Tonella[1], Filippo Ricca[1], Massimiliano Di Penta[2], Marco Torchiano[3]
[1]ITC-irst, Trento, Italy
[2]University of Sannio, Benevento, Italy
[3]Politecnico di Torino, Italy
tonella@itc.it ,ricca@itc.it, dipenta@unisannio.it, marco.torchiano@polito.it

ABSTRACT

Web application design involves at least one additional dimension over traditional software design: navigation, as supported by hyperlinks. Available design notations for Web applications offer enhanced separation of different design concerns (among which, navigation) and promise increased understandability and maintainability. However, such claims have not yet been tested in the field.

In this paper, we propose a framework for the execution of empirical studies aimed at assessing the cost-effectiveness of Web design notations. The context of the empirical studies is a typical maintenance and evolution scenario, involving activities such as program comprehension, impact analysis and change implementation. The most important obstacles and challenges in the design of such studies will be considered in this paper. We will propose counter-measures and possible mitigations for them. Finally, we will instantiate the framework into a specific empirical study that we plan to conduct in the next few months.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques; D.2.7 [**Distribution, Maintenance, and Enhancement**]:

Keywords

Empirical Studies, Web Applications, Design Notations.

1. INTRODUCTION

Web application design is a complex activity which requires the ability to deal with multiple and different kinds of concerns. A Web application is typically composed of various parts that need to be modeled at design time. Among them, the most important ones are persistent data, business logic, navigation structure, user interface. Other relevant concerns include security, transaction management, authentication. All these dimensions of a Web application must be addressed properly in the design documents.

Several design notations and methodologies have been proposed in the literature, in an attempt to provide solutions to the problems mentioned above. Among the most referenced approaches are WebML [2], UWE [6], WSDM [10], OOHDM [9], Conallen [3]. Many of these notations are extensions of UML [8]. Their most distinctive feature is typically the ability to model explicitly the navigation structure of a Web application through a dedicated model. Such a model is often accompanied by "more traditional" entity-relationship (or similar) models (for the data), static and behavioral models (e.g., class, interaction and activity diagrams) for the business logic, etc.

Separation of concerns during Web application design is clearly important during the initial development. However, it poses many problems during the maintenance and evolution phase, which actually accounts for the vast majority of an application's life cycle[4]. In fact, it is hard to keep the different views up to date and aligned. Traceability towards the implementation may be also problematic. The overlaps and interferences between different models may be hard to detect. Overall, it might be not so obvious that the benefits encountered during the initial development are kept during the evolution phase, if assessed against the associated costs (updates, alignment, traceability, etc.).

In such a context, it is extremely important to precisely understand the relative merit of the various models that have been proposed in the literature, once considered during the maintenance and evolution of an existing Web application. It might be the case that some design notations are useful mainly during the initial development, while becoming only marginally useful later, with a negative cost-benefit trade off. Others might on the contrary reveal themselves as powerful tools that can be used to tackle the typical maintenance and evolution scenarios. Gathering such knowledge is fundamental for the final user, who would be able to make an informed decision. However, no empirical study was conducted so far in this direction. In the literature, Web design methodologies are evaluated only on small examples constructed ad-hoc by the proponents or through isolated case studies, whose results cannot be usually generalized and do not provide any comparative information.

Goal	Analyze the support given by Web design notations to the comprehension and modification activities during evolution.
Null hypothesis	No significant effect on effectiveness of task execution and quality of the result.
Main factor	Design notations being validated.
Other factors	Systems, tasks, subjects and subject skills, training, tools.
Dependent variables	Knowledge acquired, capability to locate changes precisely, quality of the result.

Table 1: Template for the empirical studies.

In this paper, we propose a framework for the execution of empirical studies aimed at comparing different design notations in order to assess the support they provide to the maintenance and evolution of Web applications. The aim of the framework is to support systematic and controlled execution of experiments for the empirical validation of Web design notations. The framework specifies the high level goal and research questions of the studies, identifies the relevant factors and proposes ways to deal with the main challenges. An instance of the framework is a specific empirical study, for the assessment of specific notations in a specific evolution scenario. In this paper, an instance related to the validation of the stereotyped class diagrams (following the Conallen's notation) is presented.

Section 2 describes the framework, while Section 3 presents one example of instantiation of the framework, which is the empirical study we are going to conduct in the next few months. Conclusions and directions for future work are drawn in Section 4.

2. TEMPLATE FOR THE EXPERIMENTAL DESIGN

Table 1 summarizes the main elements of the experimental framework. Such a template follows the guidelines from well–known experimental software engineering books by Wohlin *et al.* [11] or Juristo and Moreno [5].

The general goal is quite clear: assessing Web design notations in the maintenance phase. When instantiating the framework, the general goal takes the form of a specific validation objective that addresses a specific question about the relative merit of specific notations, selected among those available in the literature.

2.1 Hypothesis

The null hypothesis is that the treatments being compared (e.g., two design notations) exhibit no significant difference. When the null hypothesis can be rejected with relatively high confidence, it is possible to formulate an alternative hypothesis, which typically admits a positive effect of one design notation in the execution of maintenance tasks. The alternative hypothesis can be further specialized according to the specific context in which it holds (see Table 3). In turn, this is characterized by the independent variables (see discussion of *other factors* below). The alternative hypothesis is formulated in terms of the main independent variable controlled in the experiment, i.e., the design notations being used.

2.2 Treatments

The treatments compared can be either two alternative web-specific design notation or a general purpose notation and a web-specific notation.

2.3 Objects

In order to support maximal internal and external validity of the studies, all the other independent variables that may affect the outcome of the study must be taken into account and possibly controlled. These include the software systems that are the object of the maintenance tasks and the tasks themselves. To mitigate the effect of these factors on the experiment's validity, the subject systems should be selected with features (size, complexity, functionality) that are typical of real Web applications. The tasks should be representative of the activities carried out by Web developers in their daily work.

2.4 Subjects

The subjects executing the maintenance tasks are another crucial factor affecting the possibility to generalize the outcome of the study. To mitigate the effects of this factor, proper training should be given to the involved subjects, so as to ensure a common, basic knowledge of the technologies involved in the experiment, as well as of the design notations being validated. Moreover, questionnaires can be used to assess the actual skills of the participating subjects and to (possibly) include them among the factors (independent variables) being considered. Such an assessment allow to properly design the experiment, ensuring a uniform distribution of subjects with high and low ability across all experiment groups. Moreover, the awareness of the subjects' ability permits to use blocking [11] when analyzing the results.

The tools provided to the subjects and the associated programming environment must be also selected carefully, so as to mimic, as much as possible, the working environment used for Web development.

2.5 Procedure and design

As discussed above, several factors affect the internal and external validity of an empirical study such as the one we are proposing. We already described ways to mitigate their effect on the generality of the results. For some of them, an additional method is counter-balancing, which can be achieved through careful design of the experimental sessions.

Table 2 shows an experimental design which balances the effects of the software system under maintenance, of the order of the treatments and of the learning curve of the involved subjects. This is achieved by dividing the subjects into four groups and involving them in at least two experimental sessions (laboratories). The order in which the systems under

Goal	Analyze the use of stereotyped UML diagrams reverse engineered from the code.
Null hypothesis 1	No significant effect on comprehension level.
Null hypothesis 2	No significant effect on impact analysis.
Null hypothesis 3	No significant effect on maintenance result.
Main factor	Stereotyped (Conallen's) UML diagrams vs. traditional UML diagrams.
Other factors	Systems (TuDu and DMS), tasks (comprehension, impact analysis and maintenance), subjects (students), training, tools.
Dependent variables	Comprehension level, accuracy of impact analysis, quality of modified code.

Table 3: Template instance for validating the use of stereotyped (Conallen) UML class diagrams in software maintenance tasks.

	Group 1	**Group 2**	**Group 3**	**Group 4**
Lab 1	Sys1-Treat1	Sys1-Treat2	Sys2-Treat1	Sys2-Treat2
Lab 2	Sys2-Treat2	Sys2-Treat1	Sys1-Treat2	Sys1-Treat1

Table 2: Experimental design.

study are presented to the subjects is reversed when considering groups 1, 2 with respect to groups 3, 4. The order of the treatments is also reversed between groups 1, 3 and 2, 4. The combination of system and treatment is completely counter-balanced, by covering every possible sequence of system and treatment.

Overall, this experimental design requires the execution of at least two experimental sessions with at least four groups of subjects. When these constraints are met, complete balancing of the order in which systems are considered and treatments are subministered is obtained.

2.6 Variables

In order to measure the effects of a treatment (design notation), metrics must be defined that allow evaluating the experimental hypotheses. For example, such metrics could capture the comprehension level reached, the ability to locate the requested change and the quality of the modified system. Questionnaires, code inspections and change impact estimates are examples of techniques that can be used to derive metrics that map to the effects to be measured.

3. INSTANTIATING THE TEMPLATE

We are planning the execution of a first empirical study that instantiates the framework described in the previous section. 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, impact analysis and maintenance. The *quality focus* is ensuring high comprehensibility and maintainability, while the *perspective* is multiple:

- **Researcher**: evaluating how effective are the stereotyped reverse engineered diagrams during maintenance.
- **Project manager**: evaluating the possibility of adopting a Web application design and reverse engineering tool in her/his organization.

3.1 Hypotheses

Since we are interested in how stereotypes affect comprehension level, impact analysis and maintenance, we formulate three different null hypotheses (and the related alternative hypotheses):

- H_{01}: 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.
  H_{a1}: 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.

- H_{02}: When doing an impact analysis task, the use of stereotyped reverse engineered class diagrams (versus non-stereotyped reverse engineered class diagrams) does not significantly affect the accuracy and the effectiveness in the execution of the task.
  H_{a2}: When doing an impact analysis task, the use of stereotyped reverse engineered class diagrams (versus non-stereotyped reverse engineered class diagrams) significantly affects the accuracy and the effectiveness in the execution of the task.

- H_{03}: When doing a maintenance task, the use of stereotyped reverse engineered class diagrams does not significantly affect the effectiveness in the execution of the task.
  H_{a3}: When doing a maintenance task, the use of stereotyped reverse engineered class diagrams significantly affects the effectiveness in the execution of the task.

3.2 Treatments

The treatment considered in this experiment is 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 the amount of improvement (if any) that can be obtained by means of Conallen's stereotypes in the maintenance and evolution phase. A similar, prelimilary study, focused on the use of stereotypes for comprehending applications related to the communication domain has been conducted by Kuzniarz *et al.* [7]. The authors showed that the use of stereotypes helped to improve the comprehension. Diagrams are reverse engineered 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.

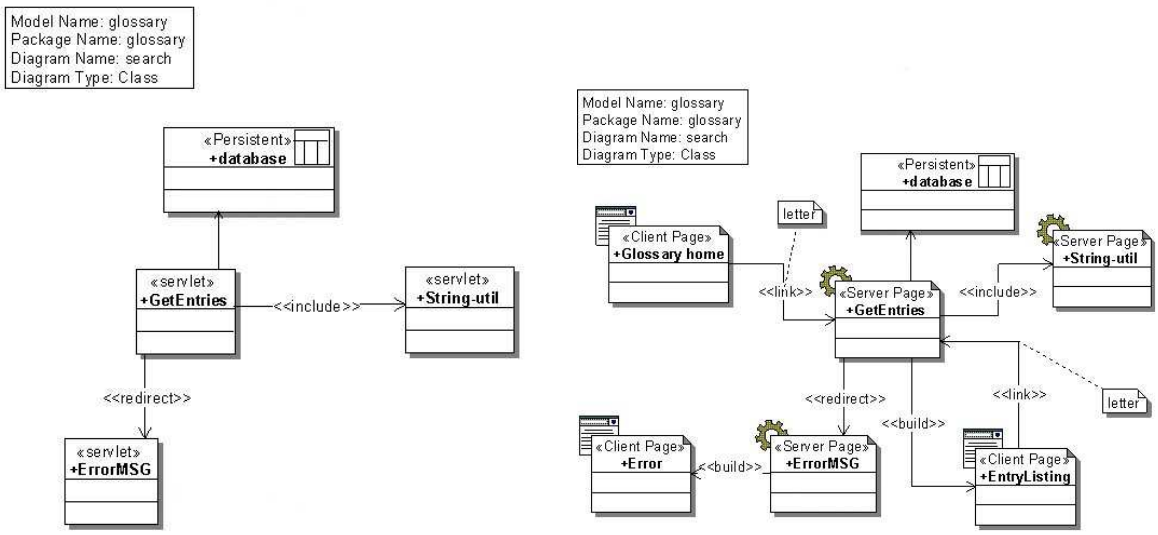

Figure 1: Basic UML class diagram (left) compared to Conallen's diagram (right).

TuDu		
	Files	**LOC**
Java	62	2929
JSP	19	1232
Total	81	4161

DMS		
	Files	**LOC**
Java	40	3731
JSP	11	1125
Total	51	4856

Table 4: Characteristics of the systems under study.

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>>`).

3.3 Objects

Two Web applications were selected for this study: *DMS* and *TuDu*. Both are small/medium size applications (see Table 4) based on the Servlet/JSP technology and downloaded from sourceforge.net. 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. The same holds for their organization and overall functioning. *TuDu*[1] is an on-line application for managing todo lists supporting cooperative work of distributed teams. It can be accessed via RSS feed. *DMS*[2] is a document management system, providing a Web centric interface to manage, access and distribute documents which are kept under version control.

3.4 Subjects

The subjects participating in the study are University students. The study will be replicated at three different sites: University of Trento, University of Sannio (Benevento) and Politecnico di Torino, in Italy. The participating students are at different levels of their course of studies, ranging from undergraduate students, to graduate and master students. Replication with students having different levels of expertise will give us the opportunity to investigate this further dimension, by comparing the results obtained at the different sites.

3.5 Procedure and design

Students will be trained on Conallen's notation, as well as all the technologies used in the target applications (e.g., Servlets/JSP). They will be involved in four 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 5, which is an instance of the counter-balanced scheme described in the previous section.

Each laboratory in the original scheme (see Table 5) is split into two (*Lab N-a, Lab N-b*, with $N = 1, 2$). The first laboratory (*Lab N-a*) consists of the execution of a comprehension task followed by impact analysis. Comprehension is driven by a request for change. Impact analysis consists

[1]http://app.ess.ch/tudu

[2]http://docmgmtsys.sourceforge.net/

	Group 1	Group 2	Group 3	Group 4
Lab 1-a	TuDu-Con	TuDu-UML	DMS-Con	DMS-UML
Lab 1-b	TuDu-Con	TuDu-UML	DMS-Con	DMS-UML
Lab 2-a	DMS-UML	DMS-Con	TuDu-UML	TuDu-Con
Lab 2-b	DMS-UML	DMS-Con	TuDu-UML	TuDu-Con

Table 5: Instantiation of the experimental design.

of an estimate of the portions of the Web application affected by the requested change, The second laboratory (*Lab N-b*) is the implementation of the change. The programming environment will be the one students are familiar with (Eclipse), with plugins supporting the design notation being validated (Conallen). The treatments indicated in Table 5 are Conallen (Con) vs. basic UML (UML).

Finally, we will ask students to fill-in a survey questionnaire (both after *Lab N-a* and *Lab N-b*) regarding the task and system complexity, the adequacy of the time allowed to complete the tasks and the usefulness of the provided diagrams.

3.6 Variables

The dependent variables of the study are:

- Comprehension level (hypothesis H_{01}).
- Capability of doing impact analysis (hypothesis H_{02}).
- Quality of the maintained code (hypothesis H_{03}).

In order to assess the effects of the treatments on the dependent variables, we will use questionnaires, test case execution and design/code inspections, and we will measure:

1. Number of correctly answered questions and time needed to answer them (both for the comprehension and for the impact analysis questionnaire).
2. Functional behavior of changed code (passed test cases).
3. Time required to implement the changes.
4. Flaws in new design (determined through inspections).
5. Code quality (determined through inspections).

4. CONCLUSIONS

The research in Software Engineering (SE) is expected to produce scientific knowledge. However, this is difficult to achieve since humans are typically in the loop of any novel SE technology. This is especially true for design notations, such as those proposed for the development of Web applications.

This work represents a first step in the direction of gathering systematic knowledge [1] about the cost-effectiveness of Web design notations. We have defined a common framework for the empirical studies focused on this topic. Then, we have instantiated the general template, obtaining the design of the first experiment that will be executed in this area. We tried to control as much as possible the factors possibly affecting the outcome of the experiment. Replication at three different sites, with different subjects, will further strengthen the results. In the design of the experiment, particular care was devoted to the balancing of the main independent variables.

A lot of future work remains to be done. First, we will actually conduct the planned experiment and replicate it at three distinct sites. We will also encourage further replications by other researchers outside the initial project team. We expect that the results of the study will provide feedback on the usefulness of different design views, according to the tasks at hand and depending on the features of the application under study. Data on the (possibly) different behaviors of subjects with different skills will be also gathered. Overall, we aim at putting the design notations proposed for Web applications in the context of a maintenance and evolution scenario, in order to assess their cost effectiveness. Replication with notations different from the one considered initially will be also fundamental to corroborate our initial findings.

5. REFERENCES

[1] V. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, July/August 1999.

[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] T. C. Jones. *Estimating Software Costs*. McGraw Hill, 1998.

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

[6] A. Knapp, N. Koch, and G. Zhang. Modeling the structure of web applications with argouwe. In *Proc. Fourth Int. Conference on Web Engineering*. Springer Verlag, July 2004.

[7] C. W. L. Kuzniarz, M. Staron. 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, Italy, 2004.

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

[9] 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.

[10] 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.

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