Using Practitioners for Assessing the Understandability of UML Statechart Diagrams with Composite States

José A. Cruz-Lemus[1], Marcela Genero[1], Sandro Morasca[2], and Mario Piattini[1]

[1] ALARCOS Research Group
Department of Information Technologies and Systems
Indra-UCLM Research and Development Institute
University of Castilla-La Mancha
Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain
{JoseAntonio.Cruz, Marcela.Genero, Mario.Piattini}@uclm.es
[2] Dipartimento di Scienze Della Cultura, Politiche e dell'Informazione
Università degli Studi dell'Insubria. Como, Italy
Sandro.Morasca@uninsubria.it

Abstract. We have carried out a family of empirical studies to investigate whether the use of composite states improves the understandability of UML statechart diagrams. Our hypothesis derived from conventional wisdom, which says that hierarchical modeling mechanisms are helpful to master a system's complexity. We carried out three studies that have gradually evolved in the size of the UML statechart models, the type of subjects (students vs. professionals), the familiarity of the subjects with the domains of the diagrams, and other factors. In this work we briefly review the first and second studies and present the third one, performed with practitioners as experimental subjects. Surprisingly, our results do not seem to show that the use of composite states improves the understandability of UML statechart diagrams.

1 Introduction

Models help us understand a complex problem and its potential solutions through abstraction. This is why software systems can benefit from using models and modeling techniques [21]. The Model-Driven Development (MDD) paradigm [1] focuses the effort of development on the design of models, rather than coding. Correspondingly, the focus of software quality assurance is shifting from system implementation towards system modeling. So, to be useful and effective, an engineering model must possess the following five key quality characteristics to a sufficient degree [21]: abstraction, understandability, accuracy, predictiveness and inexpensiveness.

In this paper, we focus on understandability because it is recognized as one of the main factors influencing maintainability, and it is well-recognized that a large part of the effort invested in the development of any software product is devoted to maintenance [19]. More specifically, we focus on the understandability of UML statechart diagrams, since UML has now become the de facto standard for modeling software systems and UML statechart diagrams have become an important technique for describing the dynamic aspects of a software system [7].

J.-L. Hainaut et al. (Eds.): ER Workshops 2007, LNCS 4802, pp. 213–222, 2007.

UML statechart diagrams are also considered as one of the most important UML diagrams [3] and should be used by practitioners as a starting point to train UML newcomers. Although the use of this kind of diagram is not critical for many systems, they are very useful for their intended purpose, have low redundancy and provide new helpful information in some systems [8], specially in real-time systems [9].

We have previously studied [5] the effect that some UML constructs may have on the understandability of UML statechart diagrams, including composite states, which are an important construct of the UML statechart diagrams metamodel [17] and a fundamental modeling abstraction mechanism. UML statechart diagrams with composite states extend finite state machines to facilitate the description of highly complex behaviors [13]. A composite state is a state that contains other states within it. Composite states are believed to be useful to model the behavior of complex classes, as we can group the simple states that are part of a larger common one, and so they help structure a system and master a system's complexity by dividing the system in smaller, less complex parts. However, it is necessary to provide empirical support to this belief to show if it is actually true and, if so, under what conditions.

Thus, based on the common use of hierarchical structures in modeling techniques for making easier to reason about a system, both for those who write the design of a system and those who read it [11, 20, 23], we hypothesized that grouping simple states that are highly related into a composite state could help improve the understandability of a diagram. We have performed a family of experiments to test this hypothesis, and in this paper we present the third empirical study corresponding to this family. We have gradually modified and improved the design and execution of these experiments, so as to alleviate some threats to the validity of the different studies that are part of it.

In Section 2 of this paper, we provide a roadmap of the family of experiments that we have performed. Section 3 explains the main features and conclusions of the two first members of the family of experiments that has been performed. Section 4 comments in detail the experimental process performed in order to carry out the third experiment of the family. Finally, section 5 summarizes the main lessons learned from this family of experiments and the future work planned to perform.

2 The Family of Experiments

An experiment can be viewed as part of common families of studies, rather than being an isolated event [2]. Common families of studies can contribute to devising important and relevant hypotheses that may not be suggested by individual experiments.

Fig. 1 shows the chronology of the family of experiments we have carried out in our study on the understandability of UML statechart diagrams.

In the initial studies (E1, R1, E2, and R2), we used students as experimental subjects. The tasks to be performed did not require high levels of industrial experience, so we thought that this experiment could be considered appropriate [2, 12]. In addition, working with students implies a set of advantages, such as the fact that the prior knowledge of the students is rather homogeneous, the availability of a large number of subjects [26], and the possibility of testing experimental design and initial hypotheses [22].

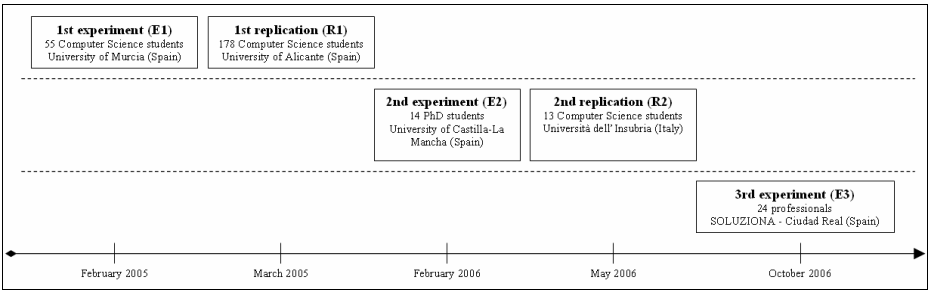

Fig. 1. Chronology of the family of experiments

The main difference between these studies and the third experiment (E3) presented in this paper, lies both in having professionals as experimental subjects, and also having renewed and improved materials and tasks.

In studies E1 and R1, we used the understandability effectiveness variable, defined as the ability to understand the presented material correctly. In studies E2, R2 and E3, we added two new variables related to the CTML, retention and transfer. Retention is defined as the comprehension of material being presented. Transfer is the ability to use knowledge gained from the material to solve related problems not directly answerable from it.

To measure all these variables, we have used three separate tests based on questionnaires. Each understandability characteristic measure is computed as the proportion of correct answers provided in each test. In this work, we will call these values UEffec for understandability effectiveness, UTrans for transfer, and UReten for retention, respectively.

We carried out the experimental processes based on well-known guidelines for empirical studies [14, 15, 28] and we used SPSS [24] to perform all the statistical analyses. In all of our studies, we set a statistical significance threshold $\alpha = 0.05$.

3 First and Second Empirical Studies

In this section, we briefly outline the main features (see Table 1 and Table 2) and conclusions reached after the performance of the two first members of the family of experiments. The complete details about them can be found in [4] and [6].

Table 1. E1 And R1 main features

Location	Universities of Murcia, Spain (E1) and Alicante, Spain (R1)
Date	February (E1) and March (R1) 2005
Subjects	55 (E1) and 178 (R1) Computer Science students
Dependent Variable	Understandability of UML statechart diagrams, measured through the measure of *UEffec*.
Independent Variables	The use or not of composite states (CS) in the diagrams.
Alternative-Hypotheses	H_{1a}: using composite states improves the *UEffect* in subjects when trying to understand an UML statechart diagram.

The materials and tasks to be performed in the first experiment and its replication (E1 and R1) were quite simple and the background knowledge of the subjects was not advanced.

The results obtained in these experiments indicated that using composite states did not improve the understandability of UML statechart diagrams. This assertion was completely against our original hypothesis, so we considered the study as preliminary and decided to perform further empirical studies for accepting or definitively reject our hypothesis.

Table 2. E2 and R2 main features

Location	Universities of Castilla-La Mancha, Spain (E2) and Insubria, Italy (R2)
Date	February (E2) and May (R2) 2006
Subjects	14 PhD students (E2) and 13 Computer Science students (R2)
Dependent Variable	Understandability of UML statechart diagrams, measured through the measures of UEffect, UTrans, and UReten.
Independent Variables	The use or not of composite states (CS) in the diagrams.
Alternative-Hypotheses	H1a: using composite states improves the UEffect in subjects when trying to understand an UML statechart diagram. H1b: using composite states improves the UTrans in subjects when trying to understand an UML statechart diagram. H1c: using composite states improves the UReten in subjects when trying to understand an UML statechart diagram.

With respect to the second experiment and its replication (E2 and R2), the materials and tasks were improved, especially with the use of the Cognitive Theory for Multimedia Learning (CTML) [16] for assessing the complete set of variables of the experimental design. More about this theory can be found in [6, 10, 16].

The results obtained with the UEffec variable agreed with the ones obtained in the E1 and R1, i.e., those UML statechart diagrams modeled without using composite states were easier to understand. Besides, the values obtained for the new variables introduced in this study, retention and transfer, were in almost all cases middle values, neither especially high nor low. This way, no strong assertions could be made about them.

So, we decided revising and improving the materials and tasks to perform in a new experiment, trying to throw light to these uncertain conclusions.

4 Third Experiment

In this section, we explain the process we followed when we carried out the third member of the family of experiments (E3).

The experiment was carried out in the facilities of Indra Company, located in Ciudad Real, Spain. Indra currently holds a top position in the market of professional

services in software, with a sales volume which is close to 800 million euros and, after a long period of expansion, the company has spread to over 28 countries in 4 different continents. The company has recently reached maturity level 3, according to the CMMi model, and the achievement of level 4 is planned for the current year 2007.

4.1 Design

In this study, we based on the design of a digital watch [27], through a model which has a size and complexity representative of a real-life case. The working hypotheses and part of the procedure to follow were similar to E2 and R2. Table 3 outlines the main features of E3.

Table 3. E3 main features

Location	Ciudad Real (Spain)
Date	October 2006
Subjects	24 professionals (12 in each group)
Dependent Variable	Understandability of UML statechart diagrams, measured through the measures of *UEffect, UTrans,* and *UReten*.
Independent Variables	The use or not of composite states (CS) in the diagrams.
Alternative-Hypotheses	H1a: using composite states improves the *UEffect* in subjects when trying to understand an UML statechart diagram. H1b: using composite states improves the *UTrans* in subjects when trying to understand an UML statechart diagram. H1c: using composite states improves the *UReten* in subjects when trying to understand an UML statechart diagram.

4.2 E3 Procedure

The experiment was divided into two sessions, in the afternoon and morning of two consecutive days.

In order to have a homogenous knowledge background in the subjects, the first session began with a seminar about "Dynamic Modeling with UML." Twenty-five professionals attended the first session, and they were provided with a summary of the main concepts of dynamic aspects in modeling in general and in UML in particular. The last part of the seminar focused on the UML statechart diagrams, although there was not an explicit mention which made the subjects guess the relation between the seminar and the subsequent experiment.

After the seminar, the instructor explained several examples related with the test that the subjects were performing in that session (Test 0). These examples consisted of questions about navigation through several statechart diagrams. Then, the subjects performed Test 0. This test was used for balancing the subjects into groups depending on their knowledge and performance.

This first session lasted approximately two hours. After it, all of Test 0 questionnaires were corrected and the subjects were grouped into two groups depending on their results.

The subjects were ordered according to the number of correct answers and the time spent on the questionnaire [18]. After that, those subjects who occupied an odd rank were assigned to Group A, and the others to Group B. Thus, we obtained two balanced groups as Table 4 shows.

Table 4. Means and standard deviations for groups after Test 0

Group	N	Correct Answers		Time	
		Mean	Std. Dev.	Mean	Std. Dev.
A	13	5.3846	1.3409	628.62	122.88
B	12	5.4583	1.0967	620.67	135.18

One of the subjects who had been assigned to Group A did not show up for Session 2. We decided to not rearrange the groups, which would now have the same number of subjects.

Session 2 was composed of three phases. In phase 1, the subjects were informed that they had been grouped depending on their performance in the test collected the day before. After that, they received one UML statechart diagram and a copy of Test 1. To avoid possible learning effects, we adopted a balanced between-subjects and blocked design, i.e. each subject was assigned only one diagram. The subjects of Group A received a diagram modelled using composite states and those in Group B received exactly the same system but modelled without using composite states.

As in E2 and R2, this test was used for measuring the UEffec of the model. The questions in Test 1 covered all the different parts of the diagram so that we could make sure that all parts of the diagram had been covered by the subjects before we removed them. This phase lasted for 25 minutes.

Then, all diagrams and tests were collected and Tests 2 and 3 were handed out. Test 2 was used for measuring the UReten variable and consisted of a fill-in-the-blanks text with 10 gaps that the subjects had to complete in order to build the text with the specifications of the system. The subjects had 15 minutes for this phase.

Test 3 was used for measuring the UTrans variable and consisted of a list of 6 tasks to perform based on the information taken from the diagram. The subjects had 35 minutes to solve them.

All the questions and tasks to perform in the different tests were exactly the same for both groups.

After the end of Test 3, all the materials were collected and the subjects were handed out a final questionnaire, to collect the subjects' impressions about the difficulty of the tests and the main positive and negative points that they had found during the experiment.

Fig. 2 graphically describes all this process.

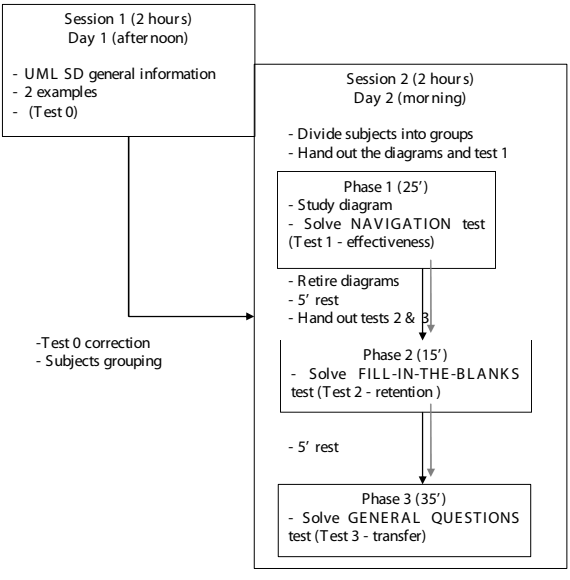

Fig. 2. E3 procedure schema

All the experimental material and data of the complete family of experiments can be found at the website http://alarcos.inf-cr.uclm.es/CSExperiments/

4.3 E3 Data Analysis and Interpretation

As in the previous studies, we carried out an analysis of the descriptive statistics of the data. Table 5 presents the descriptive statistics for the measures of the dependent variables studied in E3.

We can observe how, in this case, the results obtained for the understandability effectiveness and transfer variables are higher when the subjects worked with the diagram modelled with composite states, while the retention variable was higher in the diagram modelled without using composite states.

Table 5. Descriptive statistics for E3

CS	With (n=12)				Without (n=12)			
Stat.	Mean	Std. Dev.	Min.	Max.	Mean	Std. Dev.	Min.	Max.
UEffec	0.750000	0.1446	0.5	0.9	0.641667	0.1564	0.4	0.9
UTrans	0.369048	0.2079	0.071	0.786	0.261906	0.1677	0.036	0.571
UReten	0.775000	0.1390	0.5	1.0	0.891667	0.0515	0.8	1.0

After removing the outlier values, those which were too high and low and had fallen out of the range of typical values, we performed a t-test for testing the three hypotheses shown in Table 3 (see Table 6).

Table 6. E3 t-test results

Domain	UEffec		UTrans		UReten	
	t	sig.	t	sig.	t	sig.
Digital Watch	1.7617	0.092	1.3895	0.179	-3.6093	0.002

The results indicate that not using composite states positively affects the retention of the subjects while using composite states (weakly) improves UEffec, since the result is actually not statistically significant with our 0.05 threshold, but would be significant with a 0.1 threshold.

Regardless of the use of composite states, the subjects obtained a high level on the retention test and a low value in the transfer one. The CTML suggests that in this case, there has been only fragmented understandability but not memorization. In our opinion, the increased difficulty of the diagrams and the tasks might have produced this effect.

5 Lessons Learned and Future Work

In this work, we have presented a controlled experiment, last member of a family of empirical studies, for assessing how the use of composite states in an UML statechart diagram affects the understandability of the diagram.

We have followed the steps suggested in the Empirical Software Engineering, beginning our study with students for testing the original designs [25] and gradually improving the materials used and the experience of the subjects until performing the last study with a group of professionals.

After reviewing the obtained results in the family of experiments, we reach the following conclusions:

1. Our first idea, and the one most commonly accepted in the Software Engineering field, was that using composite states helps make a UML statechart be more understandable. The results of our empirical study do not support this idea. In most of the cases, the understandability effectiveness of the experimental subjects was better when working with systems modelled without using composite states. Nevertheless, in our last experiment, in which the subjects were professionals and hence more experienced, and the model was quite complex, the results for the understandability effectiveness were somewhat better for the subjects that used a model with composite states.
2. There is not a clear effect either in using or in not using composite states on the concepts of transfer and retention introduced by the CTML [16]. The increasing difficulty of the diagrams used and the task proposed to the subjects might have affected them, by making the diagrams more difficult to be understood.

Unexpectedly, despite performing a family of empirical studies, the evidence we collected does not support our initial hypothesis as clearly supported as we thought, and composite states do not seem to be highly useful for UML statechart diagrams understandability.

We have focused on understandability and we are planning to study some other quality aspects, such as modifiability, that could be affected by the presence of composite states. Tasks related to modifiability will probably be more complex and we will try to check whether the use of composite states makes them easier to perform.

Acknowledgements

This research is part of the MECENAS project (PBI06-0024) financed by "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" and the ESFINGE project (TIN2006-15175-C05-05) and the CALIPSO Network (TIN20005-24055-E) supported by the "Ministerio de Educación y Ciencia" (Spain). The research presented in this paper has been partially funded by the IST project "QualiPSo," sponsored by the EU in the 6th FP (IST-034763), the FIRB project "ARTDECO," sponsored by the Italian Ministry of Education and University, and the project "La qualità nello sviluppo software," sponsored by the Università degli Studi dell'Insubria.

The authors would like to thank the staff in SOLUZIONA – Ciudad Real for their time and understanding during the preparation and performance of this experiment.

References

1. Atkinson, C., Kühne, T.: Model Driven Development: a Metamodeling Foundation. IEEE Transactions on Software Engineering 20, 36–41 (2003)
2. Basili, V., Shull, F., Lanubile, F.: Building Knowledge through Families of Experiments. IEEE Transactions on Software Engineering 25, 456–473 (1999)
3. Bolloju, N., Leung, F.S.K.: Assisting Novice Analysts in Developing Quality Conceptual Models with UML. Communications of the ACM 49(7), 108–112 (2006)
4. Cruz-Lemus, J.A., Genero, M., Manso, M.E., Piattini, M.: Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 113–125. Springer, Heidelberg (2005)
5. Cruz-Lemus, J.A., Genero, M., Piattini, M.: Metrics for UML Statechart Diagrams. In: Genero, M., Piattini, M., Calero, C. (eds.) Metrics for Software Conceptual Models, Imperial College Press, UK (2005)
6. Cruz-Lemus, J.A., Genero, M., Piattini, M., Morasca, S.: Improving the Experimentation for Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In: Proceedings of 5th ACM-IEEE International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, pp. 9–11 (2006)
7. Denger, C., Ciolkowski, M.: High Quality Statecharts through Tailored, Perspective-Based Inspections. In: Proceedings of 29th EUROMICRO Conference New Waves in System Architecture, Belek, Turkey, pp. 316–325 (2003)
8. Dobing, B., Parsons, J.: How UML is Used. Communications of the ACM 49(5), 109–113 (2006)
9. Erickson, J., Siau, K.: Can UML Be Simplified? Practitioner Use of UML in Separate Domains. In: EMMSAD 2007. Proceedings of Twelfth International Workshop on Exploring Modeling Methods in Systems Analysis and Design, Trondheim, Norway, pp. 89–98 (2007)

10. Gemino, A., Wand, Y.: Evaluating Modeling Techniques based on Models of Learning. Communications of the ACM 46(10), 79–84 (2003)
11. Ghezzi, C., Jazayeri, M., Mandrioli, D.: Fundamentals of Software Engineering. Prentice Hall, Englewood Cliffs (1992)
12. Höst, M., Regnell, B., Wohlin, C.: Using Students as Subjects - a Comparative Study of Students & Proffesionals in Lead-Time Impact Assessment. In: EASE 2000. Proceedings of 4th Conference on Empirical Assessment & Evaluation in Software Engineering, Keele, UK, pp. 201–214 (2000)
13. Hu, Z., Shatz, S.M.: Explicit Modeling of Semantics Associated with Composite States in UML Statecharts. Automated Software Engineering 13(4), 423–467 (2006)
14. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers, Boston (2001)
15. Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El-Emam, K., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering 28(8), 721–734 (2002)
16. Mayer, R.E.: Multimedia Learning. Cambridge University Press, Cambridge (2001)
17. OMG, UML 2.0 - 2nd Revised Submission, Object Management Group (2003)
18. Otero, M.C., Dolado, J.J.: Evaluation of the Comprehension of the Dynamic Modeling in UML. Information and Software Technology 46(1), 35–53 (2004)
19. Pigoski, T.: Practical Software Maintenance. Wiley Computer Publishing, New York, USA (1997)
20. Pressman, R.: Ingeniería del Software. McGraw-Hill, New York (1993)
21. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software 20(5), 19–25 (2003)
22. Sjoberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V., Karahasanovic, A., Liborg, N.K., Rekdal, A.C.: A Survey of Controlled Experiments in Software Engineering. IEEE Transactions on Software Engineering 31(9), 733–753 (2005)
23. Sommerville, I.: Software Engineering. Addison Wesley, Reading (1995)
24. SPSS, SPSS 12.0, Syntax Reference Guide, SPSS Inc.: Chicago, USA (2003)
25. Tichy, W.F.: Hints for Reviewing Empirical Work in Software Engineering. Empirical Software Engineering 5, 309–312 (2000)
26. Verelst, J.: The Influence of the Level of Abstraction on the Evovability of Conceptual Models of Information Systems. In: ISESE 2004. Proceedings of 3rd International Syposium on Empirical Software Engineering, Redondo Beach, USA, pp. 17–26 (2004)
27. Webb, K.: Xholon Digital Watch Project (2006)
28. Wohlin, C., Runeson, P., Hast, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers, Boston (2000)