UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

TWEEKERKENSTRAAT 2
B-9000 GENT
Tel. : 32 - (0)9 – 264.34.61
Fax. : 32 - (0)9 – 264.35.92

WORKING PAPER

The Impact of Structural Complexity on the Understandability of UML Statechart Diagrams

José A. Cruz-Lemus [1]
Ann Maes [2]
Marcela Genero [1]
Geert Poels [2]
Mario Piattini [1]

January 2007

2007/438

[1] ALARCOS Research Group, University of Castilla-La Mancha, Spain
[2] Management Informatics Research Group, Ghent University, Belgium

The research reported in this working paper was initiated during a summer research stay of Mr. Cruz-Lemus and Dr. Genero at Ghent University (June - September 2004). The authors wish to thank Prof. Dr. Ir. Frank Gielen (Ghent University) for reviewing this working paper.

D/2007/7012/09

The Impact of Structural Complexity on the Understandability of UML Statechart Diagrams

José A. Cruz-Lemus[1], Ann Maes[2], Marcela Genero[1], Geert Poels[2] and Mario Piattini[1]

[1]ALARCOS Research Group
Department of Information Technologies and Systems
UCLM-Soluziona 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]Faculty of Economics and Business Administration
Department of Management Information, Operations Management and Technology Policy, Ghent University
Hoveniersberg 24, 9000 Gent, Belgium
{A.Maes, Geert.Poels}@UGent.be

Abstract. Given the relevance that UML models and their quality have gained in actual software development strategies, such as the Model Driven-Development (MDD), we present an empirical study about the effect that structural complexity has on the understandability of UML statechart diagrams, i.e., the diagram's ability to be easily understood. The current study is based on a family of three experiments. We have studied the data obtained in these experiments and built a preliminary understandability prediction model by means of a regression analysis using a technique specifically recommended when the data had been obtained through a repeated measures design.

1. Introduction

Paradigms such as Model-Driven Development (MDD) [1] and architectural frameworks such as the Model-Driven Architecture (MDA) [32] recognize that models are the foundation of software system development. As no system can be built on loose foundations, the focus of software quality assurance is shifting from system implementation towards system modeling.

To assure model quality, instruments are needed to evaluate and measure quality. Since the Unified Modelling Language (UML[1]) [34] became the standard for modelling software systems, a high number of quality metrics have been proposed for UML models, in particular for class diagrams [2, 9, 11, 19, 22, 28] and use case diagrams [20, 23, 30]. The development of metrics for diagrams used as behavioural models has been less emphasized. Moreover, the current metric proposals [14, 38, 39, 43] have not gone beyond the definition step. Due to the broad use of some types of behavioural diagrams, in particular statechart diagrams, there is a raising interest in controlling also the quality of these diagrams [2, 26]. In this paper we present a new set of structural complexity metrics for UML statechart diagrams and show that they can be used to evaluate a key quality, i.e., the diagram's ability to be easily understood. Diagrams that are hard to understand are difficult to analyze, modify, extend, integrate with other diagrams, or reuse. To achieve the promised benefits of MDD in terms of increased reusability and productivity, it is necessary to control model understandability. Therefore instruments are needed to measure understandability early on in the model development process. Metrics can provide such an early quality assurance instrument.

We are aware that nowadays, there are several software design tools capable to automatically produce very complex models that are syntactically correct and semantically complete and valid, but these facts do not necessarily make these models easy to understand. In this work, we assume a relationship between complexity and

[1] For this research we based on UML v.1.4 [34]. After the release of the new version of UML (UML 2) [33] we kept all the previous work as the new version does not introduce significant differences that affect to our use of UML statechart diagrams.

quality only for some types of quality, more concretely for understandability, where human users are directly and completely involved.

Our research is based on the framework defined by Briand et al. [6, 8], which is the basis for much empirical research in the area of software quality [15, 29, 37]. For example, this framework was used by Siau [41] for understanding the complexity of UML. As we can see in Figure 1, this framework hypothesizes that the structural complexity of an UML statechart diagram affects its cognitive complexity [10]. Cognitive complexity represents the mental burden of the persons who build or use models (e.g. analysts, designers, developers, testers, maintainers, users, ...). Cognitive complexity is, however, difficult to measure. To distinguish from *cognitive complexity*, we will refer to a model's collection of structural properties as *structural complexity*, which is a measurable kind of complexity. According to Systems Theory, the complexity of a system is based on the number of (different types of) elements and on the number of (different types of) (dynamically changing) relationships between them [35]. Hence, the structural complexity of an UML statechart diagram is determined by the elements that compose it.

Briand et al.'s framework hypothesizes that high cognitive complexity will result in reduced understandability which impedes the analyzability and modifiability of the model, amongst other model qualities.

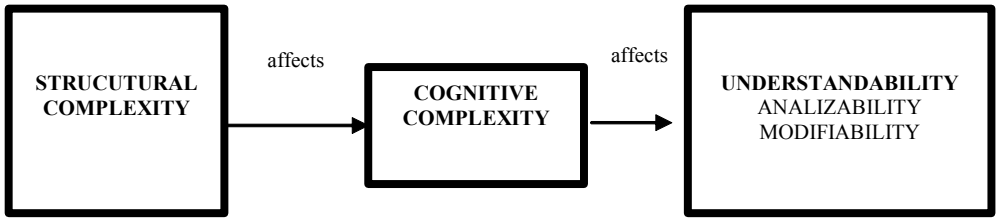


Figure 1. Relationship between structural complexity, cognitive complexity and understandability [7]

The relationship between structural complexity and external quality properties has been repeatedly demonstrated. According to Briand et al. [8], it is difficult to imagine what could be alternative explanations for these results besides cognitive complexity mediating the effect of structural complexity on software quality.

Understandability, as an external quality attribute, is hard to measure early in the modelling process. Therefore, an indirect measurement based on internal properties of the model such as the structural complexity, is required [7, 17]. In the literature, there are some interesting works related to complexity metrics [17, 22].

In section 2 we identify the UML modeling constructs that may contribute to the structural complexity of statechart diagrams and we define a metric for each of them. In section 3 these metrics are applied to a large sample of statechart diagrams in order to detect the underlying dimensions of structural complexity that they measure. Based on this understanding, a family of laboratory experiments was conducted in order to validate the metrics as understandability indicators. The design of this family of experiments is presented in section 4. In section 5 we analyze the collected data and interpret the results. In the final section 6, conclusions are presented and suggestions for further research are suggested.

2. Metrics Definition

Based on the UML meta-model [34], our measurement experience and the more commonly used elements when modelling an UML statechart diagram [16], we considered the following UML constructs as contributing to the structural complexity of UML statechart diagrams:

- **Action.** An action is a specification of an executable statement that forms an abstraction of a computational procedure that results in a change in the state of the model, and can be realized by sending a message to an object or modifying a link or a value of an attribute. In a state, we can find several types of actions: entry actions, exit actions and do/Activity actions, i.e., sequences of actions that are executed consecutively while staying in the state.
- **State.** A state is an abstract meta-class that models a situation during which some invariant condition holds. This invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions such as the process of performing some activity; that is, the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed.
- **Composite State.** A composite state is a state that contains other states vertices (states, pseudo-states, etc.). The association between the composite and the contained vertices is a composition association. Hence, a state vertex can be a part of at most one composite state.
- **Simple State.** A simple state is a state that does not have sub-states.

- **Event.** An event is the specification of a type of observable occurrence. The occurrence that generates an event instance is assumed to take place at an instant in time with no duration. Strictly speaking, the term ‘event’ is used to refer to the type and not to an instance of the type. However, on occasion, where the meaning is clear from the context, the term is also used to refer to an event instance. An event can have the association parameter, which specifies the list of parameters defined for the event.
- **Guard.** A guard is a boolean expression that is attached to a transition as a fine-grained control over its firing. The guard is evaluated when an event instance is dispatched by the state machine. If the guard is true at that time, the transition is enabled, otherwise, it is disabled. Guards should be pure expressions without side effects and have the attribute expression, which is the boolean expression that specifies the guard.
- **Transition.** A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to a particular event instance.

Based on these constructs, we defined a set of metrics for measuring structural complexity. A working hypothesis underlying the metric definition is that the more a particular construct is used when developing a statechart diagram, the more that construct adds to the structural complexity of the diagram. Hence, each metric captures the extent to which a particular construct is used in a diagram.

A brief description of the metrics is presented in Table 1. Further details about their definition can be found in [13].

Table 1. Metrics for UML statechart diagrams.

Metric	Description
NEntryA (Number of entry actions)	The total number of entry actions, i.e. the actions performed each time a state is entered.
NExitA (Number of exit actions)	The total number of exit actions, i.e. the actions performed each time a state is left.
NA (Number of activities)	The total number of activities (do/activity) in the statechart diagram.
NSS (Number of simple states)	The total number of states considering also the simple states within the composite states.
NCS (Number of composite states)	The total number of composite states.
NG (Number of guards)	The total number of guard conditions.
NE (Number of events)	The total number of events.
NT (Number of transitions)	The total number of transitions, considering common transitions (the source and the target states are different), the initial and final transitions, self-transitions (the source and the target states are the same) and internal transitions (transitions inside a state that responds to an event but without leaving the state).
CC (Cyclomatic Complexity)	It is defined as |NSS-NT+2|

These metrics were defined in a methodological way following three main steps: metric definition, theoretical and empirical validation. The theoretical validation was executed through Briand et al.'s property-based framework, which prescribes a set of intuitively derived axioms that metrics should satisfy in order to be considered as valid measures [5]. In the theoretical validation process of these metrics, we also used the Measurement Theory-based DISTANCE framework [36] for guaranteeing the construct validity of the empirical studies where these metrics were used. Through these validations, all the metrics were characterized as ratio scale metrics, which is relevant

when statistically analyzing the metrics values obtained in empirical studies. The empirical validation of these metrics is the subject of this paper.

3. Components of Structural Complexity

To substantiate our working hypothesis, we needed to investigate what kind(s) of structural complexity is (or are) measured by the metrics. As it is clear that many metric values will tend to be correlated, we do not consider each metric as measuring a different aspect of a diagram's structural complexity. We therefore first study the underlying dimensions of structural complexity captured by the metrics by employing a data reduction technique, using a sample of statechart diagrams.

In order to create summaries of the defined metrics, Principal Component Analysis (PCA) is the most commonly used technique. Principal components (PCs) are linear combinations of the standardized independent variables.

PCs are calculated as follows: the first PC is the linear combination of all standardized variables that explains a maximum amount of variance present in the data set. The second and subsequent PCs are linear combinations of all standardized variables, where each new PC is orthogonal to all previously calculated PCs, and captures the next largest amount of variance under these conditions. Usually, only a subset of all variables contributes significantly to the variance of a PC (i.e. shows a high 'loading' for that PC). In order to identify PCs and their high loading variables, we consider the rotated components. This is a technique where PCs are subjected to an

orthogonal rotation. As a result, the rotated components show a clearer pattern of loadings, where the variables either have a very low or high impact on the PC.

When applying PCA, larger samples are better than smaller samples, all other things being equal. Large samples tend to minimize the probability of errors, maximize the efficiency of population estimates, and increase the generalizability of the results. To obtain a large sample, we performed an extensive search in textbooks, journal papers and Internet sources in order to find UML statechart diagrams to include in our sample. We finally used 92 different diagrams, which is sufficient considering the sample size guidelines provided in [21]. The results of the PCA are shown in Table 2.

Table 2. PCA results

	Rotated Components		
	1	2	3
NEntryA	6.759E-02	0.892	0.222
NExitA	-7.197E-02	0.898	-8.855E-02
NA	0.302	0.216	0.704
NSS	0.808	-9.190E-03	-0.153
NCS	0.335	7.296E-02	-0.769
NE	0.876	3.729E-02	7.187E-02
NG	0.664	-1.414E-02	0.211
NT	0.975	3.743E-03	-0.111
CC	0.878	1.416E-02	-5.665E-02
	SSF	AWS	NA

Based on these results, three PCs are extracted, which jointly explain almost 75% of the variance in the data set:

- **Simple States Features (SSF),** composed by the metrics that were grouped in the first component, that is, NSS, NE, NG, NT and CC. All these metrics have in common that they explore the relationships between the different states of the diagrams and also the states themselves.

- **Activities Within States (AWS),** composed by the metrics NEntryA and NExitA, the activities performed after entering or leaving a state.
- **Number of Activities (NA).** The third component is composed only by this metric. The number of activities that a statechart diagram contains has shown to be a metric that highly affects the understandability of a diagram [13], so it is not strange that this metric has to be studied on its own.

When further investigating the relationship between structural complexity and understandability, we will work with these three components. So, we calculated the values for the components SSF and AWS as the mean of the metrics values that were included into each component.

It is important to highlight that the PCA suggested that the metric NCS, that counts the number of composite states of the diagrams, does not belong to any of the components, so we decided to discard it in this empirical validation and conducted a specific study of the effect of composite states on the understandability of the diagrams [12].

4. A Family of Experiments

In this section we will describe each step of the experimental process [42] that we followed to empirically validate the obtained components and evaluate their ability to serve as indicators for the understandability of UML statechart diagrams.

As Miller [31], Basili *et al.* [4] and Shull *et al.* [40], among others, suggested, simple studies rarely provide definite answers. Following these suggestions, we have carried out a family of experiments. We are aware that only after performing a family of experiments an adequate body of knowledge can be built to extract useful measurement conclusions regarding the use of OO design metrics to be applied to real measurement projects [4, 40].

Families of experiments promise to save preparation costs while increasing the benefits of running them. Researchers who want to participate in the family of experiments save work because they can reuse the framework and experimental material. Furthermore, reusing a framework also helps raise the quality of the studies. At the same time, individual studies possess added value when they are part of a family of experiments because they are analyzed with respect to the whole family, not only with respect to their own context. That is, families of experiments allow learning more effectively from individual empirical studies, because studies add to a body of knowledge, instead of providing information limited to one context. A higher effort is required for preparing a family of experiments but, as previously commented, the expected benefits are large and the resources investment is worth.

Our family of experiments consists of a controlled experiment and two replications of this. A descriptive graph of the chronology of the three experiments can be found in Figure 2 .

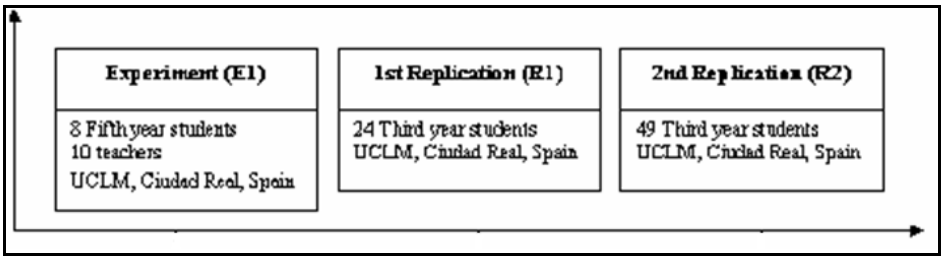


Figure 2. Chronology of the family of experiments

As most of the features are the same in the three members of the family, we will explain them together. However, we will comment on any possible difference between them.

Step 1: Definition.

Using the GQM [3] template for goal definition, the goal of the experiment and its replications is detailed in Table 2.

Table 2. Goal of the experiment.

Analyze	Structural complexity metrics for UML statechart diagrams
For the purpose of	Evaluating
With respect to	The capability of being used as indicators of the understandability of UML statechart diagrams
From the point of view of	Researchers
In the context of	Computer Science students and teachers

Step 2: Planning.

This phase consists of six different steps:

- **Context selection.** The context of the experiments was a group of teachers and undergraduate students and hence the experiment is run off-line, i.e., not in an industrial software development environment. In the first experiment (E1), the

subjects were ten teachers of the Software Engineering area and eight students enrolled in the last (fifth) year of Computer Science at the Department of Computer Science at the University of Castilla–La Mancha. In the first replica (R1), there were twenty-four students in their third-year of Computer Science and in the second replica (R2), forty-nine third-year students.

- **Subjects selection.** The subjects were chosen at our convenience. The experience of the subjects in UML statechart diagrams in E1 was average for the students, as they had already taken two complete Software Engineering courses, and it was high for the teachers, as they belonged to the Software Engineering area. In the replications, the experience of the students was lower, as they had only taken one Software Engineering course, and it had not been completed at that moment. All the teachers involved in the experiment took part voluntarily. We motivated all the students to participate in the experiments by explaining to them that similar tasks to the experimental ones could be carried out in exams or practice.
- **Variable selection.** The independent variables were the UML statechart diagrams structural complexity components SSF, AWS and NA. The dependent variable was UML statechart diagrams understandability.
- **Instrumentation.** The subjects were given twenty UML statechart diagrams, selected from different sources and related to different universes of discourse that were easy enough to be understood by each of the subjects. The structural complexity of each diagram was different, covering a broad range of the metrics values. We consider this set of twenty diagrams as a representative sample of the population of UML statechart diagrams that can be found in practice. **Fout! Verwijzingsbron niet gevonden.** shows the metrics values for the twenty UML statechart diagrams. Each diagram also had a test enclosed. It included a

questionnaire in order to evaluate if the subjects had really understood the content of the UML statechart diagrams. Each questionnaire contained four questions, which were conceptually similar and written in identical order. They inquired about navigation between states, values of variables after the execution, values for guard conditions… Furthermore, the subjects had to write down the time they started answering the questionnaire and the time they finished. The difference between these two values, expressed in seconds, is what we called *understandability time*. Diagram 20 can be found as an example in Appendix A, at the end of this document.. The dependent variable was measured by the time the subject spent answering the questionnaire attached to each diagram (understandability time) and the understandability efficiency, defined through the following formula:

understandability efficiency = correctness/understandability time *(1)*

As we can see in the formula, the understandability efficiency of a diagram is a measure that relates how correctly, i.e. correct answers vs. answered questions, and how quickly a subject understood a diagram and, in our opinion, this might be a good indicator of the actual understandability of the subjects.

- **Hypothesis formulation.** We formulated the following hypotheses:

  $H_{0,1}$: There is no significant correlation between the UML statechart diagrams structural complexity components and understandability time. $H_{1,1}$: $\neg H_{0,1}$

  $H_{0,2}$: There is no significant correlation between the UML statechart diagrams structural complexity components and understandability efficiency. $H_{1,2}$: $\neg H_{0,2}$

- **Experiment design**. We selected a within-subject design experiment, i.e., every diagram was given to every subject. However, the diagrams were ordered differently before being given to the subjects for cancelling out potential learning effects.

Table 3. Metrics and component values for each statechart diagram.

Diagram	NEntryA	NExitA	NA	NSS	NCS	NE	NG	NT	CC	SSF	AWS
1	1	1	0	3	0	6	2	5	0	16	2
2	1	0	3	4	0	6	0	7	1	18	1
3	2	0	2	4	1	4	3	7	1	19	2
4	0	0	2	4	0	11	2	9	3	29	0
5	3	2	2	4	0	13	0	10	4	31	5
6	6	6	0	6	1	12	0	13	5	36	12
7	1	0	1	5	2	6	3	10	3	27	1
8	1	0	3	5	0	12	4	13	6	40	1
9	0	0	3	5	0	8	0	11	4	28	0
10	2	1	0	4	0	6	0	6	0	16	3
11	1	2	1	6	3	12	0	17	9	44	3
12	1	1	1	3	0	5	2	5	0	15	2
13	2	1	0	2	0	4	0	4	0	10	3
14	1	1	2	3	0	8	0	9	4	24	2
15	1	0	4	9	1	11	4	13	2	39	1
16	0	0	5	9	0	23	1	23	12	68	0
17	2	0	1	5	1	6	2	8	1	22	2
18	2	0	1	12	0	23	2	24	10	71	2
19	0	1	0	2	0	5	0	5	1	13	1
20	0	0	0	5	1	11	0	12	5	23	23

Step 3: Operation.

In this phase, experimental data are collected. It includes the following activities:

- **Preparation**. In E1, the experience that the subjects had in working with UML statechart diagrams was higher than in R1 and R2, so we decided to give the subjects in the replications an intensive training session before the experiments took place. However, the subjects were not aware of which aspects we intended to study, nor were they informed about the hypotheses stated.
- **Execution**. The first experiment was performed without supervision. The subjects were given all the described materials and told to bring it back answered in one week. However, the replications were run in a two-hour session and there was an

instructor who supervised the experiment and could solve any asked doubt, although the instructor was finally not asked any question.

- **Data Validation**. Once the data were collected, we corrected them and noted down the different times and the number of answered (right and wrong) questions. From these values, we calculated the two measures of the dependent variable: the understandability time and efficiency. We also calculated automatically the metric values through a tool we had built [18]. Concerning the quality of data collecting, we used 'pencil and paper', hence data collection could be considered as critical. Supervisors did not perform any checks of the times given by students and teachers. Nevertheless, the subjects assumed the responsibility for writing down the correct times.

5. Data Analysis and Interpretation

First we tested the impact of the three extracted complexity components (SSF, AWS and NA) on the understandability of the UML statechart diagrams in terms of understandability time and efficiency through an ANOVA test. After that, we built a preliminary understandability prediction model by means of a regression analysis using a technique specifically recommended when the data had been obtained through a repeated measures design [27].

5.1 Impact of Structural Complexity on Understandability

To test the impact of structural complexity on understandability a data analysis strategy is needed that evaluates the joint effect of the three complexity components, but also

allows evaluating individual impacts and pair-wise interaction effects. To test these individual, interaction and joint effects, we considered each of the complexity components as a treatment that can be administered, independent of the administration of other treatments. To simplify the analysis, each treatment was considered at only two levels: a high level and a low level, resulting in eight possible combinations. To determine what is 'high' or 'low', the average value for each component in the set of 20 diagrams was calculated. For each diagram in the sample, if the value of a structural complexity component was over the average value, that diagram scored 'high' on the component; otherwise it scored 'low' for that component.

Next, a sub-set of the 20 diagrams representing all possible combinations of values (either high or low) for the three components was selected. This way we randomly selected a set of 8 diagrams, whose metrics values fulfilled the requirements shown in Table 4, in which, a 'H' stands for a High value of the component comparing its value with the other diagrams' values (in the full sample of 20 diagrams) and a 'L' stands for a low value.

Table 4. Diagram selection

Pattern	RS	AWS	NA	Diagram
1	H	H	H	5
2	H	H	L	6
3	H	L	H	16
4	H	L	L	20
5	L	H	H	3
6	L	H	L	13
7	L	L	H	2
8	L	L	L	19

In order to test the hypotheses, we had valid values for 84 subjects after rejecting some data for being incomplete. Since the experimental design comprised repeated

measures we performed a general linear model for repeated measures. Table 5 shows the obtained results for the main and interaction effects of each component on understandability time and efficiency when applying multivariate contrast indicators (using a cutting edge of 0.05).

Table 5. Multivariate contrasts

Effect	Understandability Time	Understandability Efficiency
SSF	0.218	0.009
AWS	0.000	0.000
NA	0.000	0.000
SSF*AWS	0.000	0.404
SSF*NA	0.000	0.000
AWS*NA	0.000	0.008
SSF*AWS*NA	0.000	0.000

As we can see in Table 5, with respect to the time, all the components and their combinations (except the SSF component) were significant. As an example, the relationship between the components SSF and AWS is described in Figure 3. When the values of the AWS components are high, the lower the values of the SSF component are, the higher the understandability time of the diagram gets. This is a relationship that we had not expected, however, for low values of the AWS components there is an inverse relationship and as the values for the SSF component decreases, the understandability time of the diagrams decreases as well.

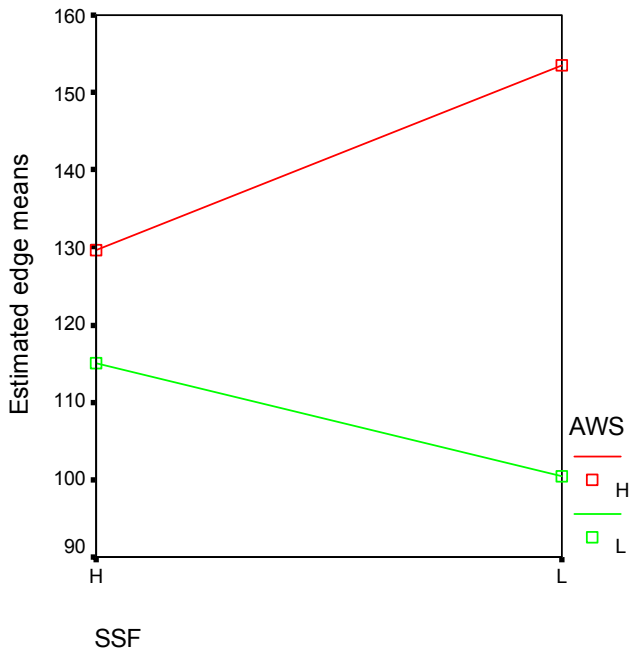


Figure 3. Estimated edge measure for understandability time (SSF*AWS)

As for the understandability efficiency, only the interaction of the components SSF and AWS is not significant, as all the rest of values are below the cutting edge (0,05). We also attach a graphical example of the relationship between two components. In this case, both for high and low values of the NA component, the lower the values of the AWS component are, the higher the understandability efficiency gets, although the pitch line improvement is more important for high values of the NA component (Figure 4).

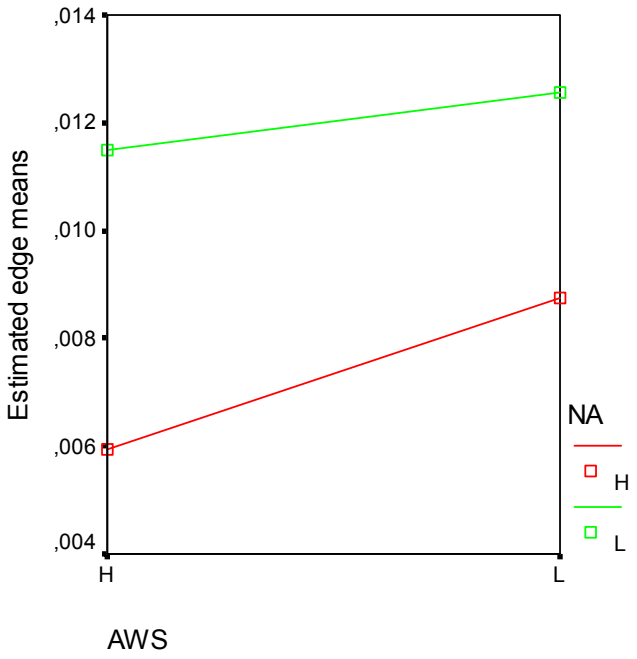


Figure 4. Estimated edge measure for understandability efficiency (AWS*NA)

5.2. Understandability model

Since we used a repeated measures design, we realized that the commonly used regression approaches were not appropriate for the data collected in the family of experiments. Hence, we used a technique outlined in [27] called *Individual Regression Equations*, and especially designed for repeated measures.

The process consists of two main steps. First, we compute separate regression equations for each subject in the family of experiments. This way each of the resulting 84 equations represents the best description for a particular subject between both the understandability time and efficiency and the set of predictor variables. At this point, the obtained regression coefficients are used to form a N*P table in which the N

subjects represent rows and the P predictor variables the columns. We do not display this table as the amount of data is excessively big.

In the second and final step of the process, we summarized over the 84 subjects each regression coefficient to see if it differs reliably from zero. This can be done with a *t* test. The results of this test are summarized in Table 6 and Table 7 for the understandability time and efficiency.

Table 6. *t* Test results for the regression models (understandability time)

Understandability Time				
Variable	CONST	NA	SSF	AWS
Mean	103.2887	8.7019	2.3591	2.0872
SE	2.9558	1.0278	0.3871	1.1511
t	34.994	8.467	6.094	1.813
Sig.	0.0000	0.0000	0.0000	0.073

Table 7. *t* Test results for the regression models (understandability efficiency)

Understandability efficiency				
Variable	CONST	NA	SSF	AWS
Mean	0.011575	-0.000813	-0.000204	-0.000273
SE	0.000335	0.000071	0.000022	0.000073
t	34.573	-11.452	-9.207	-3.720
Sig.	0.0000	0.0000	0.0000	0.0000

Since all regression coefficients were significant, we obtained two different regression equations that could be used for estimating the understandability time and efficiency based on the different complexity components:

- For the understandability time (UT):

  UT = 103.2887 + 8.7019*NA + 2.3591*SSF + 2.0872*AWS

- For the understandability efficiency (UA):

  UA = 0.011575 - 0.000813*NA – 0.000204*SSF – 0.000273*AWS

As expected, these equations show that the structural complexity of UML statechart diagrams negatively impacts the understandability efficiency and also directly affects the time necessary to get a good understanding of it. This implies that the higher the values of the complexity components are, the lower the values of the obtained efficiency will be and vice versa. Similarly, we can state that higher structural complexity values will result in higher understandability times. This gives reason to believe that these structural complexity components can serve as early indicators of model understandability.

6. Conclusions, Limitations and Future Work

Given the scarcity of metrics for measuring quality characteristics of behavioural models we have defined a set of metrics for the structural complexity of UML statechart diagrams [13]. The main focus of the current study is the empirical validation of those metrics as early understandability indicators. The empirical data necessary for performing this validation was obtained through a family of experiments

Before testing the impact of the structural complexity metrics on the understandability, we performed a PCA in order to discover the underlying dimensions of structural complexity that they measure and to reduce the number of metrics. This analysis showed that the effect of some of the metrics could be grouped into three different components:

- **Simple States Features (SSF),** composed by a set of metrics that have in common that they explore the relationships between the different states of the diagrams and also the states themselves.
- **Activities Within States (AWS),** composed by the metrics that measure the activities performed after entering or leaving a state.
- **Number of Activities (NA),** a metric measuring he total number of activities (do/activity) in the statechart diagram.

Second, we tested and confirmed the hypotheses concerning the impact of the three structural complexity components on the understandability time and efficiency. Finally, we built a preliminary understandability model using the *Individual Regression Equations* [27] technique specifically designed for data obtained through repeated measures. The resulting regression model corroborates the hypotheses that these complexity factors influence the understandability of UML statechart diagrams.

This way, when a modeller finds two or more possible semantically equivalent options for modelling a system, he should pay special attention to the use of the constructs that are part of the components described in this work, e.g., reducing the total number of activities of the diagram if possible.

Even though these findings are encouraging we consider them as preliminary because of the limitations of the current study. A first limitation is that the analysis performed here is based on correlations. We have demonstrated that structural complexity components have a statistically and practically significant relationship with the understandability of UML statechart diagrams. Such correlational relationships do

not demonstrate per se a causal relationship. They only provide empirical evidence of it. Only controlled experiments, where the components or metrics were varied in a controlled manner and all other factors were held constant, could really demonstrate causality. However, such a controlled experiment would be difficult to perform, since varying structural complexity in a system, while preserving its functionality, is difficult in practice.

Further, the generalisation capabilities of the current study seem limited. Two main threats to external validity can be identified:

- Materials and tasks used. In the experiment we tried to use statechart diagrams and tasks which can be representative of real cases, but more empirical studies taking 'real cases' from software companies must be performed in the future.
- Subjects. To solve the difficulty of obtaining professional subjects, we used teachers and students from software engineering courses. We are aware that more experiments with practitioners and professionals must be carried out in order to be able to generalize these results. However, in this case, the tasks to be performed did not require high levels of industrial experience, so experiments with students could be considered as appropriate [4, 24]. Moreover, students are the next generation of professionals, so they are close to the population under study [25].

Further research might also take into account the limitations of the current research design. Since we had a repeated measures design, the used regression technique was adapted and although this technique accurately estimates the regression

coefficients and tests the effects of each variable, it was not possible to calculate R^2 values in the regression analysis phase.

Furthermore it was surprising that composite states had not shown influence in the diagrams' understandability, so we deeply analyzed the experimental material and noticed that the selected diagrams did not use several or very complex composite states. Therefore, we think that composite states deserve further investigation. So in future research we will perform specific experiments in order to study deeply the effects of composite states on the understandability of UML statechart diagrams, through new specific experiments.

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 supported by the "Ministerio de Educación y Ciencia (Spain)" (TIN2006-15175-C05-05).

The research presented in the paper was partly performed during a summer research stay of the main author at Ghent University.

The authors would like to thank sincerely professors Félix García and Crescencio Bravo from University of Castilla-La Mancha for allowing performing the experiments with their students.

References

[1] Atkinson, C. and Kühne, T. Model Driven Development: A Metamodeling Foundation. *IEEE Transactions on Software Engineering, 20,* (2003), 36-41.

[2] Baroni, A.L., Braz, S., and Brito e Abreu, F. Using OCL to Formalize Object-Oriented Design Metrics Definitions. In *Proceedings of 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)* (Malaga, Spain, 2002). 99-106.

[3] Basili, V., Caldiera, G., and Rombach, H.D. Goal Question Metric Paradigm. *Encyclopaedia of Software Engineering, 1,* (1994), 528-532.

[4] Basili, V., Shull, F., and Lanubile, F. Building Knowledge through Families of Experiments. *IEEE Transactions on Software Engineering, 25,* (1999), 456-473.

[5] Briand, L., Morasca, S., and Basili, V. Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering, 22,* 1 (1996), 68-86.

[6] Briand, L., Wüst, J., Ikonomovski, S., and Lounis, H. Investigating Quality Factors in Object-Oriented Designs: An Industrial Case-Study. In *Proceedings of 21st International Conference on Software Engineering (ICSE 99)* (Los Angeles, USA, 1999). 345-354.

[7] Briand, L., Wüst, J., and Lounis, H., *A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study*. 1998, International Software Engineering Research Network.

[8] Briand, L., Wüst, J., and Lounis, H. Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs. *Empirical Software Engineering, 6,* 1 (2001), 11-58.

[9] Brito e Abreu, F. and Carapuça, R. Object-Oriented Software Engineering: Measuring and controlling the development process. In *Proceedings of 4th International Conference on Software Quality* (McLean, USA, 1994).

[10] Cant, S.N., Jeffery, D.R., and Henderson-Sellers, B. A Conceptual Model of Cognitive Complexity of Elements of the Programming Process. *Information and Software Technology, 7,* 351-362 (1995).

[11] Chidamber, S. and Kemerer, C. A Metrics Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering, 20,* (1994), 476-493.

[12] Cruz-Lemus, J.A., Genero, M., Manso, M.E., and Piattini, M. Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In *Proceedings of 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005)* (Mentego Bay, Jamaica, 2005). LNCS 3713, 113-125.

[13] Cruz-Lemus, J.A., Genero, M., and Piattini, M., *Chapter 7: Metrics for UML Statechart Diagrams*, in *Metrics for Software Conceptual Models*. 2005, Imperial College Press: United Kingdom.

[14] Derr, K., *Applying OMT,* SIGS Books, 1995.

[15] El-Emam, K., Benlarbi, S., Goel, N., and Rai, S. The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics. *IEEE Transactions on Software Engineering, 27,* 7 (2001), 630-650.

[16] Erickson, J. and Siau, K. Thoretical and Practical Complexity of UML. In *Proceedings of 10th Americas Conference on Information Systems* (New York, USA, 2004). 1669-1674.

[17] Fenton, N. and Pfleeger, S., *Software Metrics: a Rigurous and Practical Approach,* International Thomson Computer Press, UK, 1997.

[18] García, F., Ruiz, F., Cruz, J.A., and Piattini, M. Integrated Measurement for the Evaluation and Improvement of Software Processes. In *Proceedings of 9th European Workshop on Software Process Technology (EWSPT'9)* (Helsinki, Finland, 2003). LNCS 2786, 94-111.

[19] Genero, M., *Defining and Validating Metrics for Conceptual Models*, in *Computer Science Department*. 2002, University of Castilla - La Mancha, Spain.

[20] Genero, M., Piattini, M., and Calero, C., eds. *Metrics for Software Conceptual Models*. 2005, Imperial College Press: United Kingdom.

[21] Gorusch, R.L., *Factor Analysis,* Lawrence Erlbaum Associates, Hillsdale - New Jersey, USA, 1983.

[22] Henderson-Sellers, B., *Object-Oriented Metrics - Measures of Complexity,* Prentice-Hall, 1996.

[23] Henderson-Sellers, B., Zowghi, D., Klemola, T., and Parasuram, S. Sizing Use Cases: how to Create a Standard Metrical Approach. In *Proceedings of 8th International Conference on Object-Oriented Information Systems (OOIS 2002)* (Montpellier, France, 2002). LNCS 2425, 409-421.

[24] Höst, M., Regnell, B., and Wohlin, C. Using Students as Subjects - A Comparative Study of Students & Proffesionals in Lead-Time Impact Assessment. In *Proceedings of 4th Conference on Empirical Assessment & Evaluation in Software Engineering (EASE 2000)* (Keele, UK, 2000). 201-214.

[25] Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El-Emam, K., and Rosenberg, J. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering, 28,* 8 (2002), 721-734.

[26] Lam, V. and Padget, J. Symbolic Model Checking of UML Statechart Diagrams with an Integrated Approach. In *Proceedings of 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems* (Brno, Czech Republic, 2004). 337-347.

[27] Lorch, R. and Myers, J. Regression Analyses of Repeated Measures Data in Cognitive Research. *Journal of Experimental Psychology: Learning, Memory and Cognition, 16,* 1 (1990), 149-157.

[28] Lorenz, M. and Kidd, J., *Object-Oriented Software Metrics: A Practical Guide,* Prentice Hall, 1994.

[29] Manso, M.E., Genero, M., and Piattini, M. No-redundant Metrics for UML Class Diagram Structural Complexity. In *Proceedings of 15th International Conference on Advanced Information Systems Engineering (CAISE 2003)* (Klagenfurt, Austria, 2003). LNCS (2681), 127-142.

[30] Marchesi, M. OOA Metrics for the Unified Modeling Language. In *Proceedings of 2nd Euromicro Conference on Software Maintenance and Reengineering*1998). 67-73.

[31] Miller, J. Applying Meta-Analytical Procedures to Software Engineering Experiments. *Journal of Systems and Software, 54,* (2000), 29-39.

[32] OMG, *MDA - The OMG Model Driven Architecture,* Object Management Group, 2002.

[33] OMG, *UML 2.0 – 2nd Revised Submission*. 2003, Object Management Group.

[34] OMG, *UML Revision Task Force. OMG Unified Modeling Language Specification, v.1.4*. 2001, Object Management Group.

[35] Pippinger. Complexity Theory. *Scientific American, 238,* 6 (1978), 1-15.

[36] Poels, G. and Dedene, G., *Distance: A Framework for Software Measure Construction*. 1999, Department of Applied Economics, Catholic University of Leuven, Belgium.

[37] Poels, G. and Dedene, G. Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models. In *Proceedings of 5th European Conference on Software Maintenance and Reengineering (CSMR 2001)* (Lisbon (Portugal), 2001). 20-28.

[38] Poels, G. and Dedene, G. Measures for Assessing Dynamic Complexity Aspects of Object-Oriented Conceptual Schemes. In *Proceedings of 19th International Conference on Conceptual Modelling (ER 2000)* (Salt Lake City, USA, 2000). 499-512.

[39] Selic, B., Gullekson, G., and Ward, P., *Real-Time Object Oriented Modeling,* John Wiley & Sons, Inc., 1994.

[40] Shull, F., Carver, J., Travassos, G., Maldodano, J., Conradi, R., and Basili, V., *Replicated Studies: Building a Body of Knowledge about Software Reading Techniques*, in *Lecture Notes on Empirical Software Engineering*, N., J. and A., M., Editors. 2003, World Scientific: Singapore. p. 39-84.

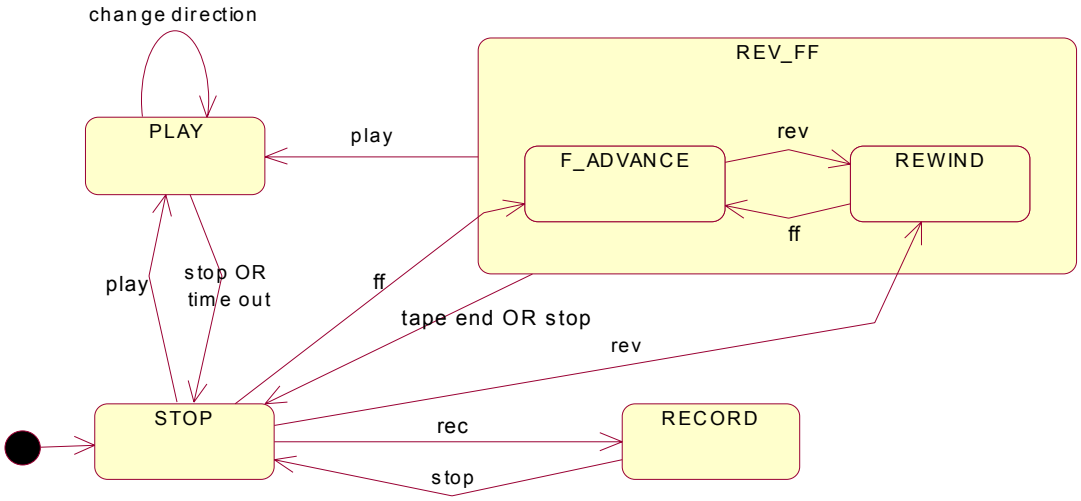
[41] Siau, K. Information Modeling and Method Engineering: a Psychological Perspective. *Journal of Database Management, 10,* 44-50 (1999).

[42] Wohlin, C., Runeson, P., Hast, M., Ohlsson, M.C., Regnell, B., and Wesslen, A., *Experimentation in Software Engineering: an Introduction.,* Kluwer Academic Publisher, 2000.

[43] Yacoub, S., Ammar, H., and Robinson, T. Dynamic Metrics for Object-Oriented Designs. In *Proceedings of 6th IEEE International Symposium on Software Metrics (METRICS 1999)* (Boca Raton, USA, 1999). 50-61.

Appendix A

Diagram 20: VCR



CHECK TIME (HH:MM:SS): _____

Please answer the following questions:

1) If while being in the state STOP the event rev occurs, which state do you get?
2) If we were in the state STOP and we have reached the state RECORD, which event would have occurred at least?
3) Which events and/or conditions would have occurred and in which order for going from the state RECORD to the state PLAY?
4) Starting at the state STOP, which state would you reach if the following sequence of events and conditions occurs?: (1) ff, (2) play, (3) change direction, (4) stop, (5) rec.

CHECK TIME (HH:MM:SS): _____