ARTICLE IN PRESS

Information Sciences xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

The impact of structural complexity on the understandability of UML statechart diagrams

José A. Cruz-Lemus [a,*], Ann Maes [b], Marcela Genero [a], Geert Poels [b], Mario Piattini [a]

[a] ALARCOS Research Group, Department of Information Technologies and Systems, UCLM-Indra Research and Development Institute, University of Castilla-La Mancha, Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain

[b] Faculty of Economics and Business Administration, Department of MIS and Operations Management, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium

ARTICLE INFO

Article history:
Received 16 November 2006
Received in revised form 10 June 2009
Accepted 25 January 2010
Available online xxxx

Keywords:
UML
Statechart diagram
Model quality
Structural complexity
Understandability
Metrics
Prediction
Empirical validation
Experiment

ABSTRACT

The effectiveness of current software development strategies, such as Model-Driven Development (MDD), depends largely on the quality of their primary artefacts, i.e. software models. As the standard modelling language for software systems is the Unified Modelling Language (UML), quality assurance of UML models is a major research field in Computer Science. Understandability, i.e. a model's ability to be easily understood, is one model quality property that is currently heavily under investigation. In particular, researchers are searching for the factors that determine an UML model's understandability and are looking for ways to manipulate these factors. This paper presents an empirical study investigating the effect that structural complexity has on the understandability of one particular type of UML model, i.e. the statechart diagram. Based on data collected in a family of three experiments, we have identified three dimensions of structural complexity that affect understandability: (i) the size and control flow complexity of the statechart in terms of features such as the number of states, events, guards and state transitions; (ii) the actions that are performed when entering or leaving a state; (iii) the sequence of actions that is performed while staying within a state. Based on these structural complexity dimensions we have built an understandability prediction model using a regression technique that is specifically recommended for data obtained through a repeated measures design. Our test results show that each of the underlying structural complexity dimensions has a significant impact on the understandability of a statechart diagram.

1. Introduction

The Model-Driven Development (MDD) [1] Software Engineering paradigm and the Model-Driven Architecture (MDA) [41] architectural framework, which provides a concrete implementation of MDD principles, 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 (software testing) towards system modelling (model verification and validation).

* Corresponding author. Address: Escuela Superior de Informática, Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain. Tel.: +34 926 295300x3748; fax: +34 926 295354.

E-mail addresses: JoseAntonio.Cruz@uclm.es (J.A. Cruz-Lemus), A.Maes@UGent.be (A. Maes), Marcela.Genero@uclm.es (M. Genero), Geert.Poels@UGent.be (G. Poels), Mario.Piattini@uclm.es (M. Piattini).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.01.026

To assure model quality, instruments are needed to measure and evaluate quality. Since the Unified Modelling Language (UML[1]) [42] became the standard language for modelling software systems, numerous quality metrics have been proposed for UML models. These proposals mainly relate to UML's structural and functionality modelling techniques, in particular to class diagrams [2,27,29], and use case diagrams [5,29,31,49]. The development of metrics for diagrams used as behavioural models has been less emphasized. Moreover, the metric proposals for the behavioural type of UML diagrams [14,15,47,54] have not gone beyond the definition step, and no validation has been performed. However, due to the widespread use of some types of behavioural diagrams [7,51], in particular statechart diagrams, there is an interest in controlling also the quality of these diagrams [2,35]. Our research aims at meeting this interest and thus focuses on instruments to measure, investigate and control the quality of UML statechart diagrams.

The framework for our research is the one defined by Briand et al. [10,12], which is the basis for much empirical research in the area of software quality [21,37,48]. For example, this framework was used by Siau [51] for understanding the complexity of UML. This framework hypothesizes that the structural complexity of a software model affects its cognitive complexity [13]. Cognitive complexity refers to the mental burden that persons (e.g. analysts, designers, developers, testers, maintainers, end-users ...) experience when building, validating, verifying or using models. Cognitive complexity is, however, difficult to measure. To distinguish from *cognitive complexity*, we will refer to a model's collection of structural properties as its *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 [45]. Hence, the structural complexity of a software model 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, adaptability and flexibility of the model, amongst other model qualities. This hypothesized relationship between structural complexity and external quality properties like understandability and modifiability has been repeatedly demonstrated [21,22,47,51]. According to Briand et al. [12], it is difficult to imagine what could be alternative explanations for these results besides cognitive complexity mediating the effect of structural complexity on quality.

Models that are hard to understand are difficult to analyze, modify, extend, integrate with other diagrams, and reuse. To achieve the promised benefits of MDD in terms of increased reusability and productivity, it is necessary to control model understandability. Understandability, per se, is not an easy-to-measure quality attribute, at least objectively and, specially, in the early stages of the software development process. Therefore, an indirect measurement (i.e. a prediction) based on the structural properties of the model is useful [11,24].

In previous work we have developed a set of structural complexity metrics for UML statechart diagrams [19,40] and we have empirically shown that some of these metrics are statistically correlated with various direct measures of understandability. However, we also noticed that most of these metrics are heavily intercorrelated and as such provide little insight into the underlying factors that determine a statechart diagram's structural complexity and impact its understandability. Furthermore, comparing the structural complexity of alternative diagrams by means of up to 10 different values is less practical than working with only a few values (e.g. is model A really better than model B if it scores better on seven out of 10 metrics whereas the other three metrics would indicate the opposite?). Also, how to control the complexity of a model if one has to look at 10 different indicators with partially overlapping behaviour?

So the goal of the current research is to reduce the number of elements to work with when indirectly measuring the understandability of an UML statechart diagram via structural complexity metrics. The data reduction technique that we use for this purpose is Principal Component Analysis (PCA). The principal components that we discovered by applying PCA on a large number of UML statechart diagrams reveal underlying dimensions of structural complexity that are captured by the metrics. Hence, they reveal the factors that impact structural complexity, and indirectly also understandability.

Based on the principal components identified, we further construct understandability prediction models using data gathered in a family of experiments. For constructing the prediction models we use a regression technique that is specifically recommended for data obtained through a repeated measures design. Our test results show that each of the underlying structural complexity dimensions has a significant impact on the understandability of a statechart diagram and that predicting understandability based on a few indicators (instead of using a comprehensive metrics suite) gives good results.

The paper is structured as follows: In Section 2 we identify the modelling constructs that contribute to the structural complexity of UML statechart diagrams and we review the previously defined metrics suite for UML statechart diagram structural complexity. In Section 3 these metrics are applied to a large sample of statechart diagrams and next PCA is applied to the obtained metric values in order to detect the underlying dimensions of structural complexity that are captured through the metrics. Based on this understanding, a family of laboratory experiments was conducted in order to develop understandability prediction models using the principal components as predictor variables. In Section 4 the design of this family of experiments is presented. The analysis of the data and the interpretation of the results is the subject of Section 5. In the final Section 6, conclusions are presented and suggestions for further research are suggested.

[1] This research is based on the ISO standard ISO/IEC 19501 for UML, i.e. version 1.4.2 [42]. Although newer versions of UML have been released (i.e. the most recent one being UML 2.2 [44]), these new versions do not introduce new features to the UML statechart meta-model that would significantly affect the treatment of UML statechart diagrams in this paper.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

2. Metrics definition

Based on the UML meta-model [42], our previous experiences with measuring UML statechart diagrams [18,40], and the most commonly used elements in UML statechart diagrams .[23], we considered the following UML constructs as contributing to the structural complexity of UML statechart diagrams:

- *State.* A state 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. Specialisations of state include composite state and simple state. A composite state is a state that contains other states (i.e. sub-states). A simple state is a state that does not contain sub-states.
- *Action.* An action is a specification of an executable statement that forms an abstraction of a computational procedure, and can be realized by sending a message to an object or by modifying a link or a value of an attribute. There exist several types of actions: entry actions, exit actions and do/Activities. An entry action is an optional action that is executed whenever the state wherein it is defined is entered regardless of the transition taken to reach that state. An exit action is an optional action that is executed whenever the state wherein it is defined is exited regardless of which transition was taken out of the state. A do/Activity is an optional activity that is executed while being in the state. The execution starts when this state is entered, and stops either by itself, or when the state is exited, whichever comes first.
- *Transition.* A transition is a directed relationship between a source state and a target state. 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.
- *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. Events trigger (compound) transitions.
- *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 an expression attribute, which is the boolean expression that specifies the guard.

Based on these constructs, we defined in [19,40] a set of nine metrics for measuring UML statechart diagram 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.

Next, a brief description of the metrics is presented. Further details about their definition can be found in [19,40].

- *NEntryA (Number of Entry Actions):* The total number of entry actions on the statechart diagram.
- *NExitA (Number of Exit Actions):* The total number of exit actions on the statechart diagram.
- *NA (Number of Activities):* The total number of do/Activities in the statechart diagram.
- *NSS (Number of Simple States):* The total number of simple 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 composite state that respond to an event but without leaving the state).
- *CC (Cyclomatic Complexity):* It is defined as $|NSS - NT + 2|$, where NSS is the Number of Simple States and NT is the Number of Transitions. The CC metric is based on McCabe's cyclomatic complexity metric [38] which equates the structural complexity of a control-flow type of diagram to the cyclomatic number of the underlying directed graph (i.e. a particular morphological property of the graph). In our adaptation, the arches of the graph correspond to the transitions in the statechart diagram and the nodes correspond to the simple states. The CC metric thus balances the number of states and state transitions. The higher the NT value relative to the NSS value, the higher the value of CC gets (because it is an absolute value), and the more complex the statechart diagram. Whereas there is a certain lower bound on the value of NT for a given NSS (because the states in a statechart diagram are connected), the CC metric captures the extra complexity that is added when increasing the number of state transitions for a diagram with a certain number of states (or as we could say, for a diagram of a given size).

All these metrics were defined in a methodological way following three main steps: metric definition and formalization, analytical/theoretical validation and empirical validation. The metrics were defined in [19] based upon an earlier, incomplete proposal [26,40]. The formalization of the metrics was performed in [17] with the Object Constraint Language (OCL) [43], an

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

expression language to be used jointly with UML diagrams, and with Maude [16], a formal language allowing for reasoning that is based on equational logic and rewriting logic. The analytical validation was executed in [28] through Briand et al.'s property-based framework [9], which contains a set of intuitively derived axioms that formalize necessary properties that metrics should satisfy in order to be considered as valid measures. In the theoretical validation process of these metrics (Cruz-Lemus et al. [19]), we also used the Measurement Theory-based DISTANCE framework [46] that defines sufficient properties 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 some of these metrics as understandability indicators was performed in [28,40].

3. Components of structural complexity

In this section we investigate the underlying dimensions of statechart diagram structural complexity that are captured by the metrics presented in the previous section. Our previous empirical studies indicate that many of the metric values tend to be correlated so we cannot conclude that the defined metrics are independent and each measure a completely different aspect of structural complexity. Consequently it is difficult to understand what structural complexity factors affect understandability and how these factors can be controlled. Further, it is not practical, nor economical to work with a large set of metrics, some of which may be redundant or at least partially overlapping.

In order to identify the different components of structural complexity that are measured by our metrics we apply Principal Component Analysis (PCA), which is a well-known statistical data reduction technique [30,34], to a set of metric data collected from a large sample of UML statechart diagrams. PCA involves a series of statistical operations (e.g. determining covariances for all pairs of observed variables) and matrix calculations (e.g. calculating eigenvectors for the covariance matrix) on a data set to transform a number of (possibly) correlated observed (i.e. measured) variables into a, usually smaller, number of uncorrelated newly identified variables. These new variables are called *principal components* (PC). The first PC accounts for as much of the variability in the data as possible, and each succeeding PC accounts for as much of the remaining variability as possible. So PCA is used to discover or reduce the dimensionality of the data set, where each identified PC represents one of the obtained orthogonal dimensions.

The extracted PCs can be seen as independent patterns of relationships between observed variables. The PCA algorithm transforms the original data so that they are expressed in terms of these patterns instead of the observed variables. As the number of PCs is equal to the number of observed variables and taken together they account for all variability in the data, no information is lost in this process. The dimensionality in a data set can be reduced by removing PCs that do not contribute much to explaining the variance in the original data set. The amount of variance accounted for by a given PC is represented by a mathematical property of the PC, the *eigenvalue* (EV). The first PC identified through PCA will have the biggest EV and subsequent PCs will have increasingly smaller EVs. Based on these EVs a cut-off for the PCs to retain can be determined. A common criterion is the Kaiser criterion that sets the cut-off at one, so only PCs with an EV greater than 1.0 are retained. A PC with an EV greater than 1.0 accounts for more variance than had been contributed by one (observed) variable, and that is why the PC is worth to be retained if reducing dimensionality is the goal.

To interpret the retained PCs, factor loadings must be examined. A factor loading is equivalent to the bivariate correlation between a PC and an observed variable. Thus a factor loading reflects the strength of the relationship between a PC and an observed variable. Interpreting a PC means identifying the variables that have high loadings for that PC and finding out what these variables have in common. One rule of thumb is to consider a factor loading high if it is greater than or equal to 0.5. Often the pattern of factor loadings is not clear, but there exist solutions for this problem in the form of rotation operations. These operations result in patterns where the observed variables have either high or low factor loadings for the extracted components.

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 [30].

We acknowledge that our sample might not be representative for the entire population of UML statechart diagrams because it did not include diagrams from real software projects. Diagrams of the kind we used are often used as training material. Obtaining real diagrams is difficult because they represent real economic value to companies and therefore, companies are reluctant to make them publicly available. We thought it was more important for the reliability of our results to obtain a large collection of diagrams, and therefore, educational models were used instead of real ones.

The values for the nine metrics presented in the previous section were calculated for each of the 92 diagrams (resulting in a data set of 828 values) and next the PCA algorithm was applied. Results in the form of a factor pattern matrix are shown in Table 1. The rows in this matrix stand for the nine metrics and the columns correspond to the extracted PCs, ordered from left to right in decreasing order of EV. There were three PCs with an EV satisfying the Kaiser criterion. To obtain a clear pattern of factor loadings a varimax rotation was used. The varimax rotation maximizes the variance of a column of the factor pattern matrix, while keeping the components uncorrelated. So the values shown in the table are the factor loadings obtained after applying the orthogonal varimax rotation. Factor loadings greater than or equal to 0.50 are shaded grey.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

Table 1
PCA results, after varimax rotation.

	Principal components		
	PC 1	PC 2	PC 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
	CFF	EEA	NA
Variance explained (cumulative)	42.28%	61.65%	74.26%

The three PCs extracted, which jointly explain almost 75% of the variance in the data set, can be interpreted as follows:

- *Control Flow Features (CFF)*, composed by the metrics with high loadings for PC 1, that is, NSS, NE, NG, NT and CC. All these metrics have in common that they measure features, like the number of states, events, guards and state transitions, which relate to the size and control flow complexity of the statechart.
- *Entry/Exit Actions (EEA)*, is PC 2 composed by the metrics NEntryA and NExitA, the actions performed when entering or leaving a state. The PCA shows that these actions form a different aspect of structural complexity than the size and control-flow complexity of the statechart diagram.
- *Number of Activities (NA)*. PC 3 is composed only by this metric. The number of do/Activities that a statechart diagram contains has shown to be a metric that highly affects the understandability of a diagram [18], and it appears to constitute an orthogonal dimension of structural complexity on its own.

Mathematically, a PC is a linear combination of the observed variables where the PCA algorithm determines the coefficients of these variables. So when measuring a set of statechart diagrams, the obtained metric values (nine per diagram) can be transformed into a smaller set (three per diagram) that still explains almost 75% of the variance in the data. Instead of using the original nine metrics, the three PCs can be used as independent variables in models that explain or predict UML statechart diagram understandability.

Furthermore, it was surprising that the use of composite states (measured through the NCS metric) did not load highly on any of the extracted structural complexity principal components, whereas we initially thought that composition could represent another dimension of structural complexity. After analyzing the experimental material we noticed that the selected diagrams did use composite states, but this use was limited and not very complex. Therefore, we thought that composite states deserved further investigation, and performed specific experiments in order to study deeply the effects of composite states on the understandability of UML statechart diagrams. The complete process and results, that indicated that using composite states did not significantly improve the understandability of simple UML statechart diagrams were published in [20].

4. A family of experiments

In this section we will describe each step of the experimental process [53] 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 [39], Basili et al. [4] and Shull et al. [50], among others, suggested, simple studies rarely provide definite answers. Following these suggestions, we have carried out a family of experiments.

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 Fig. 1.

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.

4.1. Step 1: Definition

Using the GQM [3] template for goal definition, the goal of the experiment and its replications could be stated as *Analyze* the CFF, EEA and NA principal components of UML statechart diagrams structural complexity *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 and *in the context of* Computer Science students and teachers.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

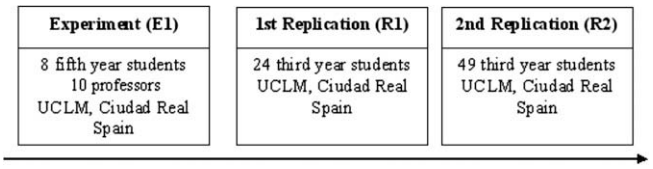

Fig. 1. Chronology of the family of experiments.

4.2. Step 2: Planning

This phase consists of six different steps:

- *Context selection.* The context of the experiments was a group of academic staff members and undergraduate students of UCLM 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 10 academic staff members of the Software Engineering area and eight students enrolled in the last (5th) year of Computer Science at the Department of Computer Science at the University of Castilla–La Mancha. In the first replica (R1), there were 24 students in their third-year of Computer Science and in the second replica (R2), 49 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 academic staff members, 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 academic staff members 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 structural complexity 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, as they cover a broad range of values for the different metrics, as shown in Table 2. The metrics values for the 20 UML statechart diagrams, were automatically calculated using GenMETRIC, a generic and extensible tool for software measurement [25].

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, etc. 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, i.e. the understandability of the diagrams, was measured by the time the subject spent answering the questionnaire attached to each diagram (*understandability time*) and by *understandability efficiency*, defined through the following formula:

$$\text{understandability efficiency}(\text{UEff}) = \text{correctness}/\text{understandability time} \quad (1)$$

As we can see in the formula, the understandability efficiency of a diagram is a measure that relates how correctly (measured by the number of right answers relative to the number of questions answered) and how quickly a subject understood a diagram. This measure has been previously employed as an understandability indicator [8].

We have tried some other formulas for measuring the understandability, e.g. using the time as stand-alone indicator, but the results were not better. Besides, there is a general agreement [6] in using understandability efficiency as a good understandability indicator

- *Hypothesis formulation.* We formulated the following hypotheses:
  $H_{0,1}$: There is no significant correlation between the UML statechart diagrams structural complexity components CFF, EEA and NA, 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 CFF, EEA and NA, 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.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

Table 2
Metrics values for each statechart diagram.

Diagram	NEntryA	NExitA	NA	NSS	NCS	NE	NG	NT	CC
1	1	1	0	3	0	6	2	5	0
2	1	0	3	4	0	6	0	7	1
3	2	0	2	4	1	4	3	7	1
4	0	0	2	4	0	11	2	9	3
5	3	2	2	4	0	13	0	10	4
6	6	6	0	6	1	12	0	13	5
7	1	0	1	5	2	6	3	10	3
8	1	0	3	5	0	12	4	13	6
9	0	0	3	5	0	8	0	11	4
10	2	1	0	4	0	6	0	6	0
11	1	2	1	6	3	12	0	17	9
12	1	1	1	3	0	5	2	5	0
13	2	1	0	2	0	4	0	4	0
14	1	1	2	3	0	8	0	9	4
15	1	0	4	9	1	11	4	13	2
16	0	0	5	9	0	23	1	23	12
17	2	0	1	5	1	6	2	8	1
18	2	0	1	12	0	23	2	24	10
19	0	1	0	2	0	5	0	5	1
20	0	0	0	5	1	11	0	12	5

4.3. 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 2-h 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 the principal component values for CFF and EEA based on the structural complexity metric values using the coefficients determined by the PCA algorithm.

5. Data analysis and interpretation

First we tested the impact of the three extracted complexity components (CFF, EEA 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 [36].

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 pairwise 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 mean 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

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

shown in Table 3, 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.

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 4 shows the obtained results for the main and interaction effects of each component on understandability time and efficiency when applying multivariate contrast indicators (using an alpha value of 0.05) obtained through an ANOVA. An ANOVA (acronym for Analysis of Variances) is a statistical technique for determining the degree of difference or similarity between two or more groups of data. It is based on the comparison of the average value of a common component [52].

The values in Table 4 are the significances of the statistical used (Miller's F in this case). A value of "< 0.001", means that we can reject the stated null-hypothesis with, at least, a 99.999% of chance of not making a mistake.

As we can see in Table 4, with respect to understandability time, all the components and their combinations (except the CFF component) had a significant impact. As for the understandability efficiency, only the interaction of the components CFF and EEA is not significant, as all the rest of values are below the cutting edge (0,05). For demonstrating interaction effects, we attach, as an example, a graphical image of the relationship between two structural complexity components, i.e. the EEA and NA components. In this case, both for high and low values of the NA component, a low value for the EEA component is associated to higher understandability efficiency. The graph shows, however, that the pitch line improvement is superior when NA is high, which indicates the existence of an interaction effect between EEA and NA (see Fig. 2).

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 [36] called *Individual Regression Equations*, which is 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 an 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 step of the process, we summarized each regression coefficient for the 84 equations to see if it differs reliably from zero. This can be done with *t* tests. The Student's *t*-test is a statistical test comparing means of normal populations with unknown standard deviations [52]. The results of these tests are summarized in Tables 5 and 6 for the understandability time and efficiency.

Tables 5 and 6 show the different results obtained after performing a t test analysis for the different components. They show the mean value, the standard deviation, the value for the t statistic and the level of significance.

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

Table 3
Diagram selection.

Pattern	CFF	EEA	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

Table 4
ANOVA results (*p* values).

Effect	Understandability time	Understandability efficiency
CFF	0.218	**0.009**
EEA	< 0.001	< 0.001
NA	< 0.001	< 0.001
CFF * EEA	< 0.001	0.404
CFF * NA	< 0.001	< 0.001
EEA * NA	< 0.001	**0.008**
CFF * EEA * NA	< 0.001	< 0.001

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

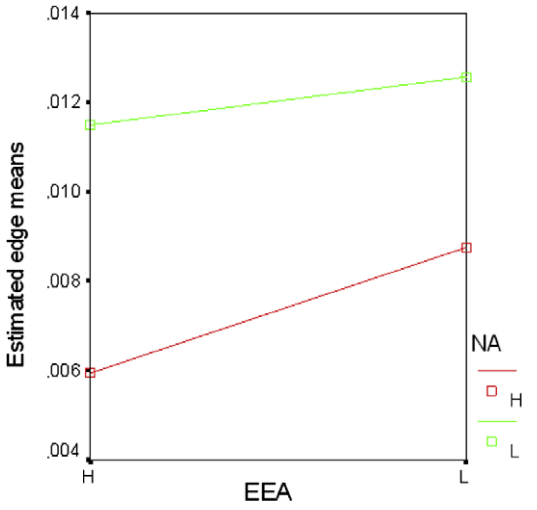

Fig. 2. Estimated edge measure for understandability efficiency (EEA * NA).

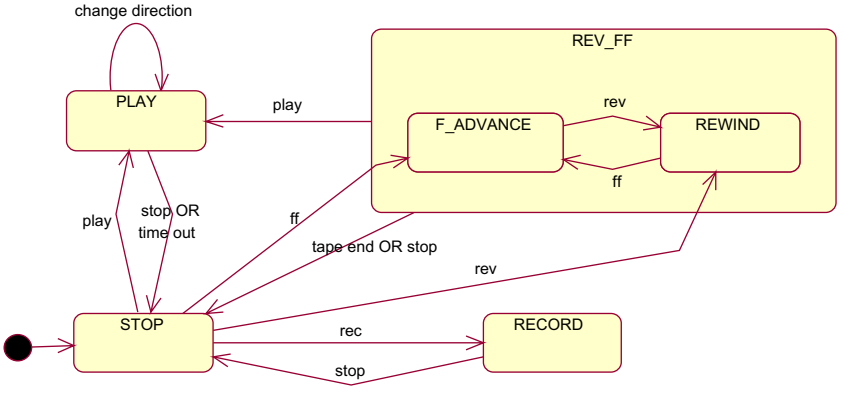

Fig. 3. Diagram 20 – VCR.

Table 5
t Test results for the regression models (understandability time).

	Understandability time			
Variable	CONST	NA	CFF	EEA
Mean	103.2887	8.7019	2.3591	2.0872
Std. Dev.	2.9558	1.0278	0.3871	1.1511
t	34.994	8.467	6.094	1.813
sig.	< 0.001	< 0.001	< 0.001	0.073

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

	Understandability efficiency			
Variable	CONST	NA	CFF	EEA
Mean	0.011575	−0.000813	−0.000204	−0.000273
Std. Dev.	0.000335	0.000071	0.000022	0.000073
t	34.573	−11.452	−9.207	−3.720
sig.	<0.001	<0.001	<0.001	<0.001

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

- For the understandability time (UT):

$$UT = 103.2887 + 8.7019^{*}NA + 2.3591^{*}CFF + 2.0872^{*}EEA$$

- For the understandability efficiency (UEff):

$$UEff = 0.011575 - 0.000813^{*}NA - 0.000204^{*}CFF - 0.000273^{*}EEA$$

These equations mean that the lower the values of NA, CFF and EEA are the lower the understandability time and efficiency will be. Besides, the higher coefficient parameter is NA, which means that UML statechart diagrams with many activities are more difficult to understand.

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 the diagrams. The higher the values of the complexity components are, the lower the efficiency demonstrated in understanding UML statechart diagrams. 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 previously defined a set of nine metrics for the structural complexity of UML statechart diagrams [19]. The main focus of the current study was the reduction of these metrics to a manageable set of structural complexity dimensions that can be used as early understandability indicators. The empirical data necessary for performing this validation was obtained through a family of experiments.

Using a sample of 92 UML statechart diagrams, we performed a Principal Component Analysis in order to discover the underlying dimensions of structural complexity that is measured by the metrics and thus to reduce the number of measurements to work with. This analysis showed that the effect of the metrics could be grouped into three different components:

- *Control Flow Features (CFF)*, composed by a set of metrics that have in common that they capture the control flow complexity of the statechart.
- *Entry/Exit Actions (EEA)*, composed by the metrics that quantify the actions 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* [36] 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. Using these equations, modellers can foresee how easy to understand their models will be and, therefore, focusing their modelling efforts in using those meta-model constructs that have a fewer negative influence on the understandability of the UML statechart diagrams.

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 must be evaluated. Two main threats to external validity can be identified:

- *Materials used.* In the experiment we tried to use statechart diagrams which are realistic (being used as training material), though not taken from real industry cases.
- *Subjects.* To solve the difficulty of obtaining professional subjects, we used academic staff members and students from software engineering courses. We are aware that more experiments with professional practitioners 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 [4,32]. Moreover, students are the next generation of professionals, so they are close to the population under study [33].

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.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

Also, as future work, we plan to perform model refactoring to UML statechart diagrams in order to check if functionality is preserved between models while the understandability of the diagrams improves.

Acknowledgements

This research is part of the following projects EECCOO (MICINN TRA2009_0074), PEGASO (MICINN/FEDER TIN2009-13718-C02-01) and the following projects financed by 'Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha': IDONEO (PAC08-0160-6141) and MECCA (PII2I09-0075-8394).

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

Appendix A. *A.1. Diagram 20: VCR (Fig. 3)*

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): _____

References

[1] C. Atkinson, T. Kühne, Model driven development: a metamodeling foundation, IEEE Transactions on Software Engineering 20 (2003) 36–41.
[2] A.L. Baroni, S. Braz, F. Brito e Abreu, Using OCL to formalize object-oriented design metrics definitions, in: 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), 2002. Malaga, Spain.
[3] V. Basili, G. Caldiera, H.D. Rombach, Goal question metric paradigm, Encyclopaedia of Software Engineering 1 (1994) 528–532.
[4] V. Basili, F. Shull, F. Lanubile, Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25 (1999) 456–473.
[5] B. Bernardez, A. Durán, M. Genero, Metrics for use cases: a survey of current proposals, in: M. Genero, M. Piattini, C. Calero (Eds.), Metrics for Software Conceptual Models, Imperial College Press, UK, 2005.
[6] F. Bodart, A. Patel, M. Sim, R. Weber, Should optimal properties be used in conceptual modelling? A theory and three empirical tests, Information Systems Research 12 (4) (2001) 384–405.
[7] N. Bolloju, F.S.K. Leung, Assisting novice analysts in developing quality conceptual models with UML, Communications of the ACM 49 (7) (2006) 108–112.
[8] L. Briand, C. Bunse, J. Daly, A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs, IEEE Transactions on Software Engineering 27 (6) (2001) 513–530.
[9] L. Briand, S. Morasca, V. Basili, Property-based software engineering measurement, IEEE Transactions on Software Engineering 22 (1) (1996) 68–86.
[10] L. Briand, J. Wüst, S. Ikonomovski, H. Lounis, Investigating quality factors in object-oriented designs: an industrial case-study, in: 21st International Conference on Software Engineering (ICSE 99), 1999. Los Angeles, USA.
[11] L. Briand, J. Wüst, H. Lounis, A comprehensive investigation of quality factors in object-oriented designs: an industrial case study, 1998, International Software Engineering Research Network.
[12] L. Briand, J. Wüst, H. Lounis, Replicated case studies for investigating quality factors in object-oriented designs, Empirical Software Engineering 6 (1) (2001) 11–58.
[13] S.N. Cant, D.R. Jeffery, B. Henderson-Sellers, A conceptual model of cognitive complexity of elements of the programming process, Information and Software Technology 7 (351–362) (1995).
[14] M. Carbone, G. Santucci, Fast && Serious: a UML-based metric for effort estimation, in: 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), 2002. Malaga, Spain.
[15] M. Cartwright, M. Shepperd, An empirical investigation of an object-oriented software system, IEEE Transactions on Software Engineering 26 (8) (2000) 786–796.
[16] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marínez-Oliet, J. Meseguer, C. Talcott, Maude 2.1.1 Manual, University of Illinois at Urbana-Champaign, 2003.
[17] J.A. Cruz-Lemus, A measurement-based approach for assessing UML statechart diagrams understandability, in: Department of Technologies and Systems Technologies, University of Castilla-La Mancha, Spain, Ciudad Real, 2007.
[18] J.A. Cruz-Lemus, M. Genero, M. Piattini, Validating metrics for UML statechart diagrams through a family of experiments, in: 1st International Workshop on Software Metrics and DASMA Software Metrik Kongress (MetriKon 2004), 2004. Königs Wusterhausen, Germany.
[19] J.A. Cruz-Lemus, M. Genero, M. Piattini, Chapter 7: metrics for UML statechart diagrams, in: Metrics for Software Conceptual Models, Imperial College Press, United Kingdom, 2005.
[20] J.A. Cruz-Lemus, M. Genero, M.E. Manso, M. Piattini, Evaluating the effect of composite states on the understandability of UML statechart diagrams, in: MODELS 2005 – LNCS 3713, 2005.
[21] K. El-Emam, S. Benlarbi, N. Goel, S. Rai, The confounding effect of class size on the validity of object-oriented metrics, IEEE Transactions on Software Engineering 27 (7) (2001) 630–650.
[22] K. El-Emam, W. Melo, The Prediction of Faulty Classes using Object-oriented Design Metrics, National Research Council of Canada, 1999.
[23] J. Erickson, K. Siau, Thoretical and Practical Complexity of UML, in: 10th Americas Conference on Information Systems, 2004. New York, USA.
[24] N. Fenton, S. Pfleeger, Software Metrics: a Rigurous and Practical Approach, International Thomson Computer Press, UK, 1997.
[25] F. García, M. Serrano, J.A. Cruz-Lemus, F. Ruiz, M. Piattini, Managing software process measurement: a metamodel-based approach, Information Sciences 177 (2007) 2570–2586.
[26] M. Genero, Defining and validating metrics for conceptual models, in: Computer Science Department, University of Castilla - La Mancha, Spain, 2002.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026

[27] M. Genero, M.E. Manso, C.A. Visaggio, M. Piattini, Building measure-based prediction models for UML class diagram maintainability, Empirical Software Engineering 12 (5) (2007) 517–549.
[28] M. Genero, D. Miranda, M. Piattini, Defining metrics for UML statechart diagrams in a methodological way, in: IWCMQ 2003 - LNCS 2814, 2003.
[29] M. Genero, M. Piattini, C. Calero (Eds.), Metrics for Software Conceptual Models, Imperial College Press., United Kingdom, 2005.
[30] R.L. Gorusch, Factor Analysis, Lawrence Erlbaum Associates, Hillsdale - New Jersey, USA, 1983.
[31] B. Henderson-Sellers, D. Zowghi, T. Klemola, S. Parasuram, Sizing use cases: how to create a standard metrical approach, in: 8th International Conference on Object-Oriented Information Systems (OOIS 2002), 2002. Montpellier, France: LNCS 2425.
[32] B. Höst, B. Regnell, C. Wohlin, Using students as subjects – a comparative study of students & proffesionals in lead-time impact assessment, in: 4th Conference on Empirical Assessment & Evaluation in Software Engineering (EASE 2000), 2000. Keele, UK.
[33] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El-Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721–734.
[34] T. Korenius, J. Laurikkala, M. Juhola, On principal component analysis, cosine and euclidean measures in information retrieval, Information Sciences 177 (22) (2007) 4893–4905.
[35] V. Lam, J. Padget, Symbolic model checking of UML statechart diagrams with an integrated approach, in: 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004. Brno, Czech Republic.
[36] R. Lorch, J. Myers, Regression analyses of repeated measures data in cognitive research, Journal of Experimental Psychology: Learning, Memory and Cognition 16 (1) (1990) 149–157.
[37] M.E. Manso, M. Genero, M. Piattini, No-redundant metrics for UML class diagram structural complexity, in: 15th International Conference on Advanced Information Systems Engineering (CAISE 2003), 2003. Klagenfurt, Austria: LNCS (2681).
[38] T.J. McCabe, A complexity measure, IEEE Transactions on Software Engineering 2 (5) (1976) 308–320.
[39] J. Miller, Applying meta-analytical procedures to software engineering experiments, Journal of Systems and Software 54 (2000) 29–39.
[40] D. Miranda, M. Genero, M. Piattini, Empirical validation of metrics for UML statechart diagrams, in: 5th International Conference on Enterprise Information Systems (ICEIS 2003), 2003. Angers, France.
[41] OMG, MDA – The OMG Model Driven Architecture, 2002. Object Management Group.
[42] OMG, Unified Modeling Language Specification, v.1.4.2. 2005. Object Management Group & International Organization for Standarization.
[43] OMG, Object Constraint Language, version 2.0. 2006. Object Management Group.
[44] OMG, Unified Modeling Language Specification v.2.1.1. 2007. Object Management Group.
[45] Pippinger, Complexity theory, Scientific American 238 (6) (1978) 1–15.
[46] G. Poels, G. Dedene, Distance-based software measurement: necessary and sufficient properties for software measures, Information and Software Technology 42 (1) (2000) 35–46.
[47] G. Poels, G. Dedene, Measures for assessing dynamic complexity aspects of object-oriented conceptual schemes, in: 19th International Conference on Conceptual Modelling (ER 2000), 2000. Salt Lake City, USA.
[48] G. Poels, G. Dedene, Evaluating the effect of inheritance on the modifiability of object-oriented business domain models, in: 5th European Conference on Software Maintenance and Reengineering (CSMR 2001), 2001. Lisbon (Portugal).
[49] M. Saeki, Embedding metrics into information system development methods: an application of method engineering technique, Lecture Notes in Computer Science 2681 (2003) 374–389.
[50] F. Shull, J. Carver, G. Travassos, J. Maldodano, R. Conradi, V. Basili, Replicated studies: building a body of knowledge about software reading techniques, in: Lecture Notes on Empirical Software Engineering, J.N. and M.A., (Ed.), 2003. World Scientific, Singapore, pp. 39–84.
[51] K. Siau, Information modeling and method engineering: a psychological perspective, Journal of Database Management 10 (44-50) (1999).
[52] B.J. Winer, D.R. Brown, K.M. Michels, Statistical Principles in Experimental Design, McGraw-Hill, 1991.
[53] C. Wohlin, P. Runeson, M. Hast, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: an Introduction, Kluwer Academic Publisher, 2000.
[54] S. Yusuf, H. Kagdi, J.I. Maletic, Assessing the comprehension of UML class diagrams via eye tracking, in: 15th IEEE International Conference on Program Comprehension (ICPC'07), 2007. Banff, Canada.

Please cite this article in press as: J.A. Cruz-Lemus et al., The impact of structural complexity on the understandability of UML statechart diagrams, Inform. Sci. (2010), doi:10.1016/j.ins.2010.01.026