Exploring the Relationships between the Understandability of Architectural Components and Graph-based Component Level Metrics

Srdjan Stevanetic
Software Architecture Research Group
University of Vienna, Austria
Email: srdjan.stevanetic@univie.ac.at

Uwe Zdun
Software Architecture Research Group
University of Vienna, Austria
Email: uwe.zdun@univie.ac.atm

***Abstract*—Architectural component models are frequently used as a central view of architectural descriptions of software systems and therefore play a crucial role in the whole development process and in achieving the desired software qualities. The components in those models represent important high level structural units that are often used to group either lower-level sub-components or classes in object-oriented design views. In this paper we present a study that examines the relationships between the effort required to understand a component, measured through the time that participants spent on studying a component, and a number of information theory based and the corresponding counting based metrics on graphs at the component level. The results show a statistically significant correlation between all of the metrics and the effort required to understand a component. In a multivariate regression analysis we obtained some reasonably well-fitting models that can be used to estimate the effort required to understand a component.**

I. Introduction

The main idea of software architecture is to concentrate on a high level view of a software system, i.e. to enable the organization of the fine-grained implementation artefacts into higher level organizational units. It drives the whole development process and plays a crucial role in achieving the desired software qualities [12]. The software architecture of the system is defined as: "the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them" [5]. Architectural component and connector models (or component models for short) are frequently used as a central view of the architectural descriptions of software systems [7]. In the context of object-oriented designs, architectural components represent important high level organization units that group classes, as well as other components, and provide one or a couple of similar system functionalities.

Architectural understanding of a software system plays a key role in managing and maintaining the overall software system. Hence, understanding of components and their interactions in component models plays a key role in supporting the architectural understanding of a software system. So far in the software architecture literature we find only a very few studies that provide empirical evidence regarding the architectural understandability or the measurement of architectural understandability (see e.g. [9], [8]). To the best of our knowledge, there is no existing empirical study on the understandability of architectural component models (the two previously cited studies [9], [8] examine understandability at the package level).

In this paper we present a study that aims to examine the relationships between the effort required to understand a component measured through the time that participants spent on studying a component and a number of information theory based and the corresponding counting based metrics on graphs at the module (component) level. The metrics have been previously defined by Allen et al. [2], [3]. The subjects of the study were 49 master students. The software system to be studied by the participants was the Soomla Android store Version 2.0, an open source framework for supporting virtual economy in mobile games. In order to answer 4 true/false questions for each component the participants had to fully understand the functionalities of each component by exploring the relationships (together with their roles) between the classes within each component and the relationships that those classes have with the classes outside of the given component.

The results of our analysis show a statistically significant correlation between all of the metrics and the effort required to understand a component. In a multivariate regression analysis we obtained some reasonably well-fitting models that can be used to estimate the effort required to understand a component. Improving our knowledge of how to create understandable architectural components helps to improve the quality of component models and the software they represent [4].

This paper is organized as follows: In Section II, we briefly discuss the related work. In Section III we describe the study design. Section IV describes the statistical methods we applied and the analysis of our data. In Section V we discuss the threats to validity of the study. In Section VI we conclude and discuss future directions of our research.

II. Related Work

Model understandability has been studied by a number of authors in the field of data models. In that context, model understandability has been defined as the ease with which the model can be understood [15]. Moody proposes three metrics for model understandability: the model user rating of model understandability, the ability of users to interpret the model correctly, and the model developer rating of model understandability [15]. In the work by Patig [16] the variables and tasks that have been proposed by cognitive psychology

or applied in computer science to test understandability are extracted. All variables have been theoretically justified by the authors that used them. With regard to work by Patig we measured the correctness of the answers and the time that participants spent on resolving the questions.

Many different software metrics for measuring the system's architecture, components as its constituting parts, and structures similar to architectural component models, such as other higher-level software structures (packages, graph-based structures) have been proposed. Metrics related to components and the corresponding architectures [18], [19] measure size and different dependencies of individual components but also the complexity of the whole architecture when all the components and their interactions are taken into account. Different authors have proposed different package level metrics [8], [9], [14]. Graph-based metrics measure different interactions between the nodes in the graph [13], [3]. All the mentioned metrics can be applied or can be more-less easily adapted to be applicable for the component models. However, none of the metrics is empirical validated regarding understandability of architectural components or architectural component models so far.

III. Empirical study description

For the study design we have followed the experimental process guidelines proposed by Kitchenham et al. [10] and Wohlin et al. [21]. The former was primarily used in the planning phase of the study while the later was used for the analysis and the interpretation of the results.

A. Goals

The main idea of this study is to explore the relationships between the understandability of the components in architectural component models and graph based component level metrics. Two sets of metrics are used in the study. The first set encompasses information theory based metrics (built on the very well notion of the Shannon entropy) at the module level of a graph abstraction of a software system defined by Allen [2]. The second set encompasses the corresponding counting based metrics that are defined by Allen at al. in a later research work [3] [1]. The given metrics measure component's size, complexity, coupling, cohesion and length (build on the notion of size applied to paths). We choose the given sets of metrics because: 1) they are easily adapted to be applicable to the component models (i.e. have a sufficient level of abstraction), 2) they measure different attributes that are very closely related to the understandability concept (size, complexity, coupling, and cohesion), and 3) they can probably be combined using the adequate statistical analysis for creating the appropriate understandability prediction models. In our previous study, we showed the usefulness of some package level metrics adapted from the work by Martin [14] in assessing the understandability of the components [20]. In this study we examine the given graph based metrics and compare the results in Section IV-C to our previous results. We do not explain the details about the metrics definitions (that also require the appropriate description of the corresponding notation) in this article because of space limitations. Comparing to the work by Allen et al., in our study nodes represent the source code classes and edges represent the relationships between the classes. Classes are grouped into components (which correspond to the modules in the work by Allen et al.).

B. Variables

We differentiate two groups of variables in our study. The first group of variables was collected from the participants of the study while the second group of variables was collected from the studied system. The first group of variables includes three independent variables related to demographic information: programming experience, commercial programming experience, and experience in programming computer games. It also includes two more variables, time required to study a component and the percentage of the correct answers on the study questions. The time variable is measured by the time that the participants spent on studying each component, and it is used to measure the effort required to understand a component (dependent variable). The percentage of the correct answers is introduced to help in estimating the understandability effort in case that the participants do not spend enough time to deeply study all the components in the system in order to achieve a high percentage of the correct answers. Namely, if the participants do not spend enough time on studying the given component, the percentage of the correct answers will probably decrease for that component; so there is a dependency relation between these two variables. Therefore, the percentage of the correct answers can assist in estimating the time required to fully study the given component (i.e., to achieve 100 % of the correct answers), which can be used as an indicator of the effort required to fully understand a component [2]. In other words in order to estimate the effort required to fully study a component the value for the percentage of the correct answers can be replaced by the constant value of 100 % in the obtained prediction models (see Section IV-C). From the statistical point of view predicting the effort needed for the 100 % correctness is not the most realistic because there are not so many data points available for that. More data points are available for the 75 % correctness for example and the most realistic would be to find the appropriate value between these two values (75% and 100%). Nevertheless in our models we use 100% because the differences in the prediction in our case are negligible. The second group of variables are related to the metrics that we aim to explore. They are all independent variables.

Regarding the size metrics, we expect that the bigger the size of a module the more effort is required to understand it. Regarding the length metrics we expect the same behaviour as for the size metrics because they are based on the notion of size. Regarding the complexity metrics that consider the amount of information in relationships we expect that if there are more relationships between the nodes in a module themselves and between the nodes in a module and the outside world (higher complexity) the more effort is required to understand it. Regarding the coupling metrics that are based on the

[1] In our study we consider ordinary undirected graphs, and the given counting based metrics are applied in that context unlike those in the work by Allen et al. [3] that are applied to hyper-graphs. Ordinary graphs contain edges that connect only two graph nodes unlike hyper-graphs where one edge can connect more than two nodes.

[2] Predicting the percentage of the correct answers variable is also possible but since our focus is on estimating the time variable that is used as an indicator of the effort required to understand a component we consider the percentage of the correct answers as an auxiliary variable for the time prediction as it is explained in Section III-B

definition of complexity considering only intermodule edges we expect the same behaviour as for the complexity metrics. Regarding the cohesion metrics that consider the amount of information in intramodule relationships with respect to a complete module graph (i.e., all nodes are connected to all other nodes which represents the "most cohesive" module) we expect that more cohesive modules requires less effort to be understood because they contain closely related services and functionalities.

The dependent variables together with their scale types, units, and ranges are shown in Table I. The independent variables are shown in Table II.

Description	Scale type	Unit	Range
Time	Ratio	Minutes	Positive natural numbers including 0

TABLE I. DEPENDENT VARIABLES

Description	Scale type	Unit	Range
Programming experience	Ordinal	Years	4 categories: [0,1),[1-3),[3-7), >=7
Commercial programming experience	Ordinal	Years	4 categories: [0,1),[1-3),[3-7), >=7
Experience in programming computer games	Ordinal	Years	4 categories: [0,1),[1-3),[3-7), >=7
Size (inform. and count.)	Ratio	bit/node	Positive real/integer numbers incl. 0
Complexity (inform. and count.)	Ratio	bit/edge	Positive real/integer numbers incl. 0
Coupling (inform. and count.)	Ratio	bit/edge	Positive real/integer numbers incl. 0
Length (inform. and count.)	Ratio	bit/node	Positive real/integer numbers incl. 0
Cohesion (inform. and count.)	Ratio	-	Positive real/rational numbers incl. 0
Percentage of the correct answers	Ratio	-	[0,100]%

TABLE II. INDEPENDENT VARIABLES

C. Hypotheses

Based on previous considerations we formulate the following set of hypotheses:

H_{01}: There is a significant positive correlation between the size metrics of a component and the effort required to understand a component.

H_{02}: There is a significant positive correlation between the complexity metrics of a component and the effort required to understand a component.

H_{03}: There is a significant positive correlation between the coupling metrics of a component and the effort required to understand a component.

H_{04}: There is a significant negative correlation between the cohesion metrics of a component and the effort required to understand a component.

H_{05}: There is a significant positive correlation between the length metrics of a component and the effort required to understand a component.

D. Study design

1) Subjects: The subjects of the study are the 49 master students of the Advanced Software Engineering (ASE) lecture at the University of Vienna in the Winter Semester 2013.

2) Objects: The software system to be studied by participants was the Soomla Android store[3] Version 2.0, an open source framework for supporting virtual economy in mobile games. The choice of using this particular system is motivated by the following factors: 1) the Soomla Android store is a free open source system, which enables us to conduct the study and disseminate its results, 2) it is used in real-world games and therefore it has industrial relevance, 3) it is written in the Java programming language with which the participants were sufficiently familiar, and 4) the source code of the Soomla Android store adheres to coding standards and is rather easy to understand for most potential subjects (the source code classes are also well designed in terms that there are no big deviations in their sizes, i.e. each of them provides one or a couple of similar functionalities).

3) Instrumentation: The following instruments were used to carry out the study.

a) Architectural documentation about the Soomla Android store version 2.0: The documentation describes the conceptual architecture and lists technologies and frameworks used in the implementation. Besides text, a UML component diagram is used to illustrate the components in the system, and their inter-relationships in parts of the architecture. Participants were also provided with the set of traceability links, showing the relations between architectural components and their realized code classes.

b) Browser-based source code access: Browser-based access to the source code of Soomla Android store was provided in a Lab environment on prepared computers. All source code classes were grouped in the corresponding components so that the participants can easily study the components in the system by studying their realized source code classes.

c) A questionnaire to be filled-in by the participants during the experiment: On the first page of the questionnaire, the participants had to rate their experience. The subsequent pages contain the understanding questions. In order to answer the questions correctly the participants had to fully understand the functionalities of each component by exploring the relationships (together with their roles) between the classes within each component and the relationships that those classes have with the classes outside of the given component. There were 4 true/false questions for each component. In the case of bigger components, answering the questions requires studying of more classes than in the case of smaller components. The tasks were randomized so that 7 different random combinations of the 7 components were generated and randomly assigned to the participants. The randomization was used to ensure that we get the more/less balanced data for all the components in terms of equalizing the confounding factors such as possible fatigue effects or the lack of time needed to complete all the tasks.

We also provided a table where the participants had to enter the time slots during which they studied each of the components. Each time slot contains a start and stop time, indicating the time when the participants started studying the given component and the time when they finish it, respectively. There were more time slots in case the participants wanted to

[3]see: http://project.soom.la/

study the component more than one time. The time is written in the format *hour : minute*. The document that contains the explained instruments can be found on the following web address [4].

E. Execution

1) Preparation: As it is explained in Section III-D the study was conducted at the University of Vienna, Austria in the context of a lecture on Advanced Software Engineering. The total time limit for the whole study was 1.5 hours.

2) Data collection: According to the experience of the participants we can say that the participants have medium to high programming experience (most of them have [3,7) and more than 7 years of programming experience). Many of them have industrial programming experience, while only a very few have experience in game programming. All participants had knowledge about software development and software architecture, as well as of software traceability.

The mean, the median and the standard deviation of the time and the percentage of the correct answers variables collected from the participants are shown in Figure 1. The participants with [0,1) years of programming experience were excluded from the consideration. Some participants did not write the time they spent on studying the particular components (they did not write the start time or the stop time or both of them) and those participants were excluded from the consideration for those particular components. Some of them spent very short time in studying the components which is not enough[5] and can just introduce bias in the results and those participants were also excluded from the consideration for the given component.

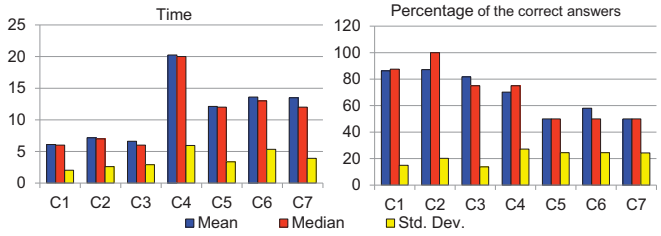

Fig. 1. Time and percentage of the correct answers – descriptive statistics

The data related to the given metrics are shown in Table III. The metrics are automatically calculated from the corresponding graph abstractions of the studied system. The accuracy of the metrics calculations is tested on the examples provided in the work by Allen [2].

By looking at Figure 1 we can observe that the time needed to study the first three components is significantly decreased comparing to the other components in the system. This is an expected result because those 3 components are smaller in terms of the number of classes they contain. It can be also observed that the time the participants spent on studying the components *C5*, *C6* and *C7* is decreased compared to the component *C4*. The percentage of the correct answers (see Figure 1) for the components *C5*, *C6* and *C7* is also decreased compared to the component *C4* and the smaller components *C1*, *C2* and *C3*. Even though it is realistic to expect that the percentage of the correct answers for the bigger components decrease on average (because of the higher amount of information that needs to be studied which increases the probability of missing some information), it seems also that the participants needed a bit more time for studying the components *C5*, *C6* and *C7* (or at least for studying the component *C7* which has the same number of classes as the component *C4*) in order to achieve the higher percentage of the correct answers. Based on this consideration we obtained some prediction models for the time variable that use both the given set of metrics and the percentage of the correct answers.

Component level metrics	Size		Complexity		Coupling		Cohesion		Length	
	Info	Count	Info	Count	Info	Count	Info	Count	Info	Count
Security (C1)	11.23	2	62.72	8	44.15	7	1.00	1.00	11.23	2
CryptDecrypt (C2)	28.07	5	138.2	14	54.82	9	0.49	0.50	16.84	3
PriceModel (C3)	16.84	3	66.72	7	31.34	5	0.61	0.67	16.84	3
GooglePlayBilling (C4)	61.76	11	222.3	19	42.11	7	0.27	0.27	28.07	5
StoreController (C5)	44.92	8	205.7	25	125.2	20	0.31	0.33	16.84	3
DatabaseServices (C6)	44.92	8	298.2	29	101.5	16	0.47	0.46	16.84	3
StoreAssets (C7)	61.76	11	274.3	24	64.73	10	0.40	0.39	22.46	4

TABLE III. COMPONENT LEVEL METRICS

IV. ANALYSIS

For statistical analysis of the obtained data, we used the programming language R [17].

A. Testing Hypotheses

1) Testing the Normality: As the first step in analysing the data, we test the normality of the data by applying the Shapiro-Wilk normality test in R. After applying the normality tests we found that all variables do not fit the normal distribution (all p-values are much less than 0.05; that is the null hypothesis can be rejected). Based on that we decided to pursue the non-parametric Spearman rank correlation test with our data in the next step of the analysis.

2) Testing the Correlation Between the Variables: In order to test our hypotheses the Spearman rank correlation test is used with a level of significance $\alpha = 0.05$. It examines whether there is a linear correlation between the tested variables.

Table IV shows the Spearman's coefficients and the corresponding p-values between the time variable and the given metrics. Given the results from the undertaken correlation analysis, it has been demonstrated that all hypotheses of our study ($\mathbf{H_{01}}$, $\mathbf{H_{02}}$, $\mathbf{H_{03}}$, $\mathbf{H_{04}}$, and $\mathbf{H_{05}}$) are supported.

If we take a closer look at Table IV we observe that the correlation coefficients for the size and the length information theory based metrics are equal to the coefficients of the corresponding counting based metrics. Furthermore, the coefficients are very similar for the complexity, coupling, and cohesion metrics. It suggests that there is a high correlation between the given two metrics sets (information theory based and counting based metrics).

Regarding the information theory based size metric we observe the whole system graph (all nodes and edges in the system) and conclude that all nodes' patterns are unique because each edge has exactly two end points and there are no disconnected nodes (nodes without incident edges). In that

[4]https://swa.univie.ac.at/soomla-architectural-components/

[5]We expect that at least 30 seconds is needed to study each class in the component.

	Size (1), Complexity (2), Coupling (3), Cohesion (4), Length (5)	
	Information theory based	Counting based
Time	**1:** r=0.7691 **p-value<2.2e-16**	**1:** r= 0.7691 **p-value<2.2e-16**
	2: r=0.6818 **p-value<2.2e-16**	**2:** r= 0.6103 **p-value<2.2e-16**
	3: r=0.2303 **p-value=0.0002**	**3:** r= 0.3329 **p-value=4.8e-08**
	4: r=-0.771 **p-value<2.2e-16**	**4:** r=-0.770 **p-value<2.2e-16**
	5: r=0.6807 **p-value<2.2e-16**	**5:** r= 0.6807 **p-value<2.2e-16**

TABLE IV. THE SPEARMAN CORRELATION COEFFICIENTS

case the information theory based size metric is proportional to the number of nodes in a module, i.e. counting size [2].

For the complexity, coupling and cohesion metrics it is possible to prove (using some approximations of the metrics formulas) for the given system that there is a linear dependency between the information theory based metrics and the corresponding counting based metrics (we do not show the derivations of that proof because of space limitations). Please note that the mentioned proof is specifically related to the studied system. For some other systems the correlation can be violated to a greater/lesser extent depending on the relationships between the graph nodes.

B. Collinearity Analysis

To create prediction models that can be used to predict the time variable, first we have to conduct a collinearity analysis between the variables that can be the possible predictors of the time variable and to exclude those variables that are highly correlated with other possible predictors. We consider the given two sets of metrics (information theory based metrics and counting based metrics) as two separate sets of predictors because we already showed that they are highly correlated between each other in our case. Therefore, all possible predictors include either all information theory based metrics or all counting based metrics and the percentage of the correct answers (according to the discussion in Section III-B). Accordingly, the Condition Number (CN) and the Variance Inflation Factor (VIF) values for the metrics were calculated. If the VIF values are higher than 10, multicollinearity is strongly suggested. The acceptable values for the condition number are the values less than 30 (a threshold suggested in the literature [6]).

In principle the predictors with the highest VIF values greater than 10 are step-by-step excluded from the set of all predictors. In our case we have to exclude two predictors and they are either the Size and the Length metrics or the Complexity and the Length metrics (for the set of information theory based metrics). Using the same reasoning for counting based metrics three final sets of possible predictors are obtained (two of them exclude the same metrics as those sets obtained for information theory based metrics and the third one excludes the Size and the Cohesion metric).

C. Multivariate Regression Analysis

Multivariate regression analysis is performed to construct different multivariate regression models that can be used to predict the time variable. We used the Mallows' Cp calculation to create reasonably fitting models that prevent over-fitting of the data [11]. All the models that have $Cp \leq p$ (p - number of predictors including the constant predictor, if present) must be considered reasonably good fits.

There are totally 14 models (8 models for the counting based metrics and 6 models for the information theory based metrics) that fit the explained criteria ($Cp \leq p$). The accuracy of the predicted models is checked using different results in R. The residuals are checked to follow the normal distribution. The influential points are the points whose removal will cause a large change in the fit. Those points can be detected using the Cook's distance contour lines. If some points have a distance larger than 1, it suggests the presence of a possible outlier or a poor model. The obtained models do not have such influential points. The coefficient of determination (R^2) is used to describe how well the regression fits a set of data. The Mean Magnitude of Relative Error (MMRE) and prediction at level 0.25 (Pred(0.25)) measures are calculated as de facto standard and commonly used measures for the evaluation of the accuracy of the predicted models.

For all obtained models adjusted R^2 is in the range [84,88] %, MMRE is in the range [35,39] %, and Pred(0.25) is in the range [42,50] %. The given models' parameters are very similar to those obtained in our previous work [20], where we examined some package level metrics adapted from the work by Martin [14]. Both sets of models can account for around 88 % of the variance in the data (maximum adjusted $R^2 \approx 88\%$). Therefore, both metrics sets can be used as predictors of the effort required to understand a component with the same efficiency. In this study we obtained more models (14 models comparing to 3 models in the previous one) that can be considered as reasonably good fits because more sets of possible predictors are used.

Beside the Mallows' Cp calculation we also tested our models using the 10-fold cross-validation technique which is also useful for overcoming the problem of over-fitting [1]. The 14 models obtained in the analysis using the Mallows' Cp criterion performed the best in the cross-validation analysis as well with respect to the adjusted R^2 value. The values for MMRE and Pred(0.25) are almost the same as those obtained in the analysis using the Mallow's Cp. It confirms the validity and the results of our previous analysis using the Mallows' Cp criterion. The results of the Mallow's Cp and the cross-validation analyses are not shown because of space limitations.

Using the obtained prediction models we can calculate the time variable in order to achieve the maximum percentage of the correct answers (100 %) which represents the effort required to fully understand a component that is actually the measure of the component's understandability. Therefore, by replacing the percentage of the correct answers variable with the constant value of 100% (some other more precise values can be used with respect to the discussion in Section III-B) we obtain the models for the prediction of the component's understandability based on the given set of metrics.

V. VALIDITY EVALUATION

In this section we discuss the threats to validity of our study and how we tried to minimize them:

a) Conclusion validity: The conclusion validity defines the extent to which the conclusion is statistically valid. The study is limited to the small-size dataset that consists of 7 components due to the limited time of the study session. We

plan to increase the number of studied components and to investigate other penitential metrics in our future work.

b) Construct validity: The construct validity is the degree to which the independent and the dependent variables are accurately measured by the appropriated instruments.

A possible threat to validity might be the measuring of the time variable. The participants could have forgotten to write the time right before they start and right after they finish studying the components which represents a threat to the accuracy of the time variable. In order to reduce that threat we wrote a reminder before each component to remind the participants to write the stop time in the previously studied component if they forgot it and the start time for the given component they intend to study as the next one.

c) External validity: The external validity is the degree to which the results of the study can be generalized to the broader population under study. A threat to external validity might be the size of the classes in a component. As it is mentioned in Section III-D2 there are no big deviations in the sizes of the classes in the Soomla system, i.e. each of them provides one or a couple of similar functionalities. In the general case, there could be some classes that are much bigger than other classes in the system and in that case the number of classes in a component will not be an appropriate measure of its size. This case actually is not in accordance with good design principles, i.e. the big classes should be divided into smaller classes that encompass one or a couple of similar functionalities but this case can be examined in terms of which deviations in the classes' size are acceptable and do not violate the performed analysis.

VI. Conclusions and future work

In this paper we present the study that aims to examine the relationships between a number of information theory based and the corresponding counting based metrics on graphs at the component level of a software system on one side and the effort required to understand a component on the other side. The metrics used in the study are previously defined in the work by Allen et al. [2], [3] and they measure size, complexity, coupling, cohesion and length. The effort required to understand a component is measured through the time that the participants spent on studying each of the components. The results of the analysis show statistically significant correlation between all of the metrics and the effort required to understand a component. In a multivariate regression analysis we obtained some reasonably well-fitting models that can be used to estimate the effort required to understand a component. Improving our knowledge of how to create understandable architectural components helps to improve the quality of the architectural component models and the software they represent. In our future work we plan to study more components and to investigate more metrics and their relationships to the understandability of components and architectural component models.

Acknowledgement

This work was supported by the Austrian Science Fund (FWF), Project: P24345-N23. We thank Dr. Nina Senitschnig from the Department of Statistics and Operations Research, for valuable suggestions and help related to the statistical analysis pursued in the study.

References

[1] Cross Validation techniques in R: A brief overview of some methods, packages, and functions for assessing prediction models.

[2] E. B. Allen. Measuring graph abstractions of software: An information-theory approach. In *IEEE METRICS*, pages 182–. IEEE Computer Society, 2002.

[3] E. B. Allen, S. Gottipati, and R. Govindarajan. Measuring size, complexity, and coupling of hypergraph abstractions of software: An information-theory approach. *Software Quality Control*, 15(2):179–212, June 2007.

[4] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock. Quality attributes. Technical report CMU/SEI-95-TR-021, Software Engineering Institute, 1995.

[5] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[6] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity (Wiley Series in Probability and Statistics)*. Wiley-Interscience.

[7] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Boston, MA, 2003.

[8] M. O. Elish. Exploring the relationships between design metrics and package understandability: A case study. In *ICPC*, pages 144–147. IEEE Computer Society, 2010.

[9] V. Gupta and J. K. Chhabra. Package coupling measurement in object-oriented software. *J. Comput. Sci. Technol.*, 24(2):273–283, Mar. 2009.

[10] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on*, 28(8):721–734, Aug. 2002.

[11] M. Kobayashi and S. Sakata. Mallows' cp criterion and unbiasedness of model selection. *Journal of Econometrics*, (3):385–395.

[12] F. Losavio, L. Chirinos, N. Lvy, and A. Ramdane-Cherif. Quality characteristics for software architecture. *Journal of Object Technology*, 2(2):133–150, 2003.

[13] Y. Ma, K. He, D. Du, J. Liu, and Y. Yan. A complexity metrics set for large-scale object-oriented software systems. In *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, CIT '06, pages 189–, Washington, DC, USA, 2006. IEEE Computer Society.

[14] R. C. Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

[15] D. L. Moody. Metrics for evaluating the quality of entity relationship models. In *Proceedings of the 17th International Conference on Conceptual Modeling*, ER '98, pages 211–225, London, UK, UK, 1998. Springer-Verlag.

[16] S. Patig. A practical guide to testing the understandability of notations. In *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling - Volume 79*, APCCM '08, pages 49–58, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.

[18] K. Sartipi. A software evaluation model using component association views. In *IWPC*, pages 259–268, 2001.

[19] S. Sengupta, A. Kanjilal, and S. Bhattacharya. Measuring complexity of component based architecture: a graph based approach. *SIGSOFT Softw. Eng. Notes*, 36(1):1–10, Jan. 2011.

[20] S. Stevanetic and U. Zdun. Exploring the relationships between the understandability of components in architectural component models and component level metrics. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, EASE 2014, London, UK, 2014. ACM Computer Society.

[21] C. Wohlin. *Experimentation in Software Engineering: An Introduction: An Introduction*. The Kluwer International Series in Software Engineering. Kluwer Academic, 2000.