Exploring the Understandability of Components in Architectural Component Models using Component Level Metrics and Participants' Experience

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.at

***Abstract*—Architectural component models play a crucial role in achieving the desired software quality, as understandability of components and their interactions plays a key role in supporting the architectural understanding of a software system. In this article, we extend our previous studies on component models understandability. Our extensions study hierarchical understandability metrics, the impact of personal factors of participants like experience and expertise, and the combinations of both personal factors and the metrics (the previously studied and the newly introduced). The subjects of the study had to fully understand the functionalities of a number of components of an open source system by exploring the relationships of the components' classes. Our results provide evidence that the hierarchical understandability metrics are significantly better in predicting the understandability effort than the models obtained in our previous studies or the models that include just the participants' experiences. The participants' experience plays an important role in the prediction but the obtained prediction models are not as accurate as the models that use the component level metrics.**

***Keywords*-architectural component models; understandability; software metrics; empirical study;**

I. Introduction

Software architecture focusses on a high level view of a software system, and it 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" [4]. It represents a key artefact that affects all other activities such as design and implementation and plays a crucial role in achieving the desired software qualities [18].

Architectural component and connector models (or component models for short) are frequently used as a central view of the architectural representation of the system [8]. With respect to object-oriented designs, architectural components group classes, as well as other components, and provide a set of closely related system's functionalities.

Understanding architectural components and their interactions is essential for architectural understanding and plays a key role in managing and maintaining the overall software system. To the best of our knowledge, there is no empirical evidence related to the understandability of architectural component models.

In our previous work [23, 24] we presented studies that examine the relationships between the effort required to understand an architectural component, measured through the time that participants spent on studying a component, and a number of metrics calculated from the studied system. Those metrics include some package-level metrics adapted from the work by Martin [19] (studied in [24]) and a number of information theory based metrics and the corresponding counting based metrics on graphs at the component level defined by Allen et al. [2, 3] (studied in [23]). In this study we further elaborate on the understandability concepts by: 1) studying a hierarchical quality metrics model [14], 2) studying the impact of personal factors (i.e. participants' experience and expertise) on the obtained results, 3) combining and comparing both personal and model factors (metrics) in order to check if the newly introduced concepts and metrics improve the previously obtained prediction models. The input data used for the analyses in all three mentioned articles (including this one) is taken from the same empirical study.

The results of our analysis show that the hierarchical understandability metrics are significantly better in predicting the understandability effort than the metrics obtained in our previous work. However, the prediction models that use hierarchical understandability metrics are not significantly different in prediction from the models that combine both model related metrics (the previously studied and the newly introduced metrics) and the participants' experience. The participants' experience is important and can predict a significant amount of variance in the data but the obtained models are not as accurate as the models that use the metrics related to the software model itself (concretely the hierarchical understandability metrics).

This article is organized as follows: In Section II, we 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. Finally, in Section VI we conclude and discuss future directions of our research.

II. Related Work

So far we find only a very few studies related to the empirical evidence on the architectural understandability. One of them examines the influence of package coupling on the understandability of the software systems [13], while another one examines the relationships between some package-level metrics and package understandability [10]. None of the studies examines the understandability of architectural components.

There exist plenty of software metrics for measuring the system's architecture, architectural components, and other high level software artefacts and structures (see [25] for an overview). All these metrics can be adapted to be applicable for the component models but none of them is empirically evaluated with respect to understandability of those models. In our empirical studies we try to evaluate the usefulness of some of the mentioned metrics for assessing the understandability of architectural components.

The work in the field of process model related metrics emphasizes the importance of model characteristics for assessing model understandability. Such metrics measure structural properties of a process model, motivated by prior work in software engineering related to lines of code, cyclomatic number, or object-oriented metrics [20, 7, 11]. Similar to the study by Reijers and Mendling [22] who investigate the impact of personal and model related factors on understandability of process models we investigate the impact of those two kinds of factors on understandability of architectural component models. They show that expert modellers perform significantly better and that the complexity of the model affects understanding. They find that personal factors (theoretical knowledge, practical experience, educational background) have a stronger explanatory power (in terms of adjusted R^2) than model related factors but they kept the size of the models constant. In our study we also find that participants' experience is important as well as model related metrics but in contrast to the work by Reijers and Mendling we find that model related metrics have a significantly stronger explanatory power and even alone can be used for a prediction, i.e., combining them with the experiences does not produce a stronger explanatory power. The size is taken into account in our study in contrast to the study by Reijers and Mendling. Also, all our participants are students, and we do not consider experts from industry as it is the case in the study by Reijers and Mendling.

III. Empirical study description

In the planning phase of our study we have followed the experimental process guidelines proposed by Kitchenham et al. [16]. The guidelines proposed by Wohlin et al. [26] have been used for the analysis and the interpretation of the results.

A. Goals

As mentioned above, in this study we examine the efficiency of the hierarchical understandability metrics proposed in the work by Hwa et al. [14] as well as the participants' experience in predicting the effort required to understand an architectural component and try to improve the prediction capabilities of our previously obtained prediction models.

The metrics from the hierarchical understandability assessment model are shown in Table I together with the properties they measure. The following notation is used: MD – the set of all modules; C – the set of all classes; $C(md)$ – the set of classes in a module md; $rel_c(c_1, c_2)$ – it is TRUE if a class c_1 depends on a class c_2 by method calls, data reference or inheritance relationships, otherwise FALSE; $rel_{md}(md_1, md_2)$ – it is TRUE if there exist any classes $c_1 \in md_1$ and $c_2 \in md_2$ so that $rel_c(c_1, c_2) = TRUE$, otherwise FALSE; $MD_a(md)$ – the set of ancestor modules of a module md in the module hierarchy. Please note that modules in the work by Hwa et al. correspond to architectural components in our case.

In this paragraph we provide explanations for the metrics definitions and briefly discuss the relationships between the metrics and the understandability of modules (components). The MSC (Module Size in Classes) metric is defined as the total number of classes in a module. Typically bigger modules are more difficult to understand. The NAC (Number of API Classes) metric is defined as the number of classes in a module, which are accessed by other classes in other modules. An encapsulated module limits the access to its elements by offering a small number of interfaces to other modules which facilitates understanding. The DMC (Direct Module Coupling) metric is defined as the number of modules to which a given module is directly coupled. The higher the coupling of a module the more difficult it is to understand it. The NDC (Number of Disjoint Clusters) metric is defined as the number of disjoint clusters of classes in a module, and it measures the cohesion of a module. The classes that interact cohesively to provide a focused service are directly or indirectly connected to each other. Therefore, the higher the NDC metric of a module, the more difficult it is to understand it. In our case there are no disjoint clusters in any of the components and therefore all components have an NDC metric equal to 0. The CRW (Cohesion by Rest of World) is defined as the number of classes outside a module that are commonly shared by the classes in a module. Namely the classes in a module with cohesive functionalities have relationships to the similar set of classes outside a module. Therefore the higher the metric, the easier it is to understand a module. Finally, the DMH (Depth in Module Hierarchy) metric is defined as the depth of a module in a module hierarchy. Considering modules as components this metric is not directly applicable in our case because our components contain classes located in different modules/packages and therefore it is not possible to determine the depth of a component in a module/package hierarchy. But similarly we can find an average depth in a hierarchy for all classes in a component with respect to the location of the class in a module/package hierarchy.

Metric's Name	Metric's Definition	Measured Property
Module Size in Classes (MSC)	$\mathrm{MSC}(md) = \lvert C(md)\rvert$	Design Size/Complexity
Number of API Classes (NAC)	$NAC(md) = \lvert\{c_1 \in C(md) \mid \exists c_2 \in C(md2)[rel_c(c_2, c_1) \wedge md_2 \in MD \wedge md_2 \neq md]\}\rvert$	Encapsulation
Direct Module Coupling (DMC)	$DMC(md) = \lvert\{md_2 \in MD \mid rel_c(md, md_2) \vee rel_c(md_2, md), md \neq md_2\}\rvert$	Coupling
Number of Disjoint Clusters (NDC)	$NDC(md) = \left\lvert\left\{cl \subseteq C(md) \middle\vert \forall c_i \in cl \left[\left(\lvert cl\rvert = 1 \vee \exists c_j \in cl \left(rel_c(c_i, c_j) \vee rel_c(c_j, c_i)\right)\right) \wedge \nexists c_k \in C\left[\left(c_k \notin cl \wedge \left(rel_c(c_i, c_k) \vee rel_c(c_k, c_i)\right)\right)\right]\right]\right\}\right\rvert$	Cohesion
Cohesion by Rest of World (CRW)	$CRW(md) = \sum \frac{\lvert SRC(c)\rvert}{\lvert \bigcup_{c \in C(md)} SRC(c)\rvert}, SRC(c) = \left\{c_2 \in C \middle\vert \left(c_2 \in \left(C - C(md)\right)\right) \wedge \left(rel_c(c, c_2) \vee rel_c(c_2, c)\right)\right\}$	Cohesion
Depth in Module Hierarchy (DMH)	$DMH(md) = \lvert MD_a(md)\rvert$	Abstraction

Table I
Definitions of the hierarchical understandability metrics (adapted from [14])

B. Variables

The first set of variables that are collected from the participants includes 7 variables, from which 5 are independent variables related to the participants' demographic information: programming experience, Java programming experience, commercial programming experience, experience in programming computer games, and Android programming experience, and the remaining two are the time required to study a component and the percentage of the correct answers on the study questions. The variables related to the

participants' demographic information are chosen to capture the knowledge relevant for understanding the studied system.

The time variable is used to measure the effort required to understand a component, and it represents a dependent variable. The percentage of the correct answers variable is introduced to help in estimating the time variable, i.e. the required effort, in the case that the participants do not spend enough time to fully study the given components in order to achieve a high percentage of correctness. Namely, there exist a dependency between the time and the percentage of the correct answers variables because if the participants spend less time on studying a component, the percentage of the correct answers will probably decrease. Therefore with the help of the percentage of the correct answers variable we can estimate the time required to fully understand a given component, i.e., to achieve 100 % of the correct answers. If we replace the value for the percentage of the correct answers in the obtained prediction models (see Section IV-B) with the constant value of 100 %, the effort required to fully understand a component is obtained, that further depends only on other factors included in the model. This study design aspect is discussed in more detail in our previous study (see [24]) with additional explanations using the obtained results.

The second set of variables are related to the metrics that we aim to explore (see Table I), and they are calculated form the studied system. All the metrics are treated as independent variables.

The dependent variable and its scale type, unit, and range is shown in Table II, while the independent variables together with their scale types, units, and ranges are shown in Table III.

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

Table II
DEPENDENT VARIABLE

Description	Scale type	Unit	Range
Prog (programming exp.)	Ratio	Years	Positive rational numbers incl. 0
JProg (java programming exp.)	Ratio	Years	Positive rational numbers incl. 0
CProg (commercial programming exp.)	Ratio	Years	Positive rational numbers incl. 0
GProg (games programming exp.)	Ratio	Years	Positive rational numbers incl. 0
AProg (android programming exp.)	Ratio	Years	Positive rational numbers incl. 0
Answers (percentage of the correct answers)	Ratio	-	[0,100]%
MSC (Module Size in Classes)	Ratio	Class	Positive integer numbers incl. 0
NAC (Number of API Classes)	Ratio	Class	Positive integer numbers incl. 0
DMC (Direct Module Coupling)	Ratio	Module	Positive integer numbers incl. 0
NDC (Number of Disjoint Clusters)	Ratio	-	Positive integer numbers incl. 0
CRW (Cohesion by Rest of World)	Ratio	Class	Positive rational numbers incl. 0
DMH (Depth in Module Hierarchy)	Ratio	-	Positive integer numbers incl. 0

Table III
INDEPENDENT VARIABLES

C. Hypotheses

Regarding our hypotheses we expect that the given hierarchical understandability metrics can be used as good predictors of the understandability effort. Furthermore, we expect that the participants' experience is significant and important in predicting the understandability effort but we do not expect that it can capture the variability of the measured understandability effort as good as the metrics related to the component model itself. Finally, by combining both the previously studied metrics (the graph-based and the package-level metrics) and the newly introduced metrics (Table I) with the participants' experience we expect that more efficient prediction models can be obtained compared to those that consider separately the component level metrics and the participants' experience.

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

Hypothesis ($\mathbf{H_1}$): The hierarchical understandability metrics introduced in the work by Hwa et al. [14] can be successfully utilized to construct reasonably well-fitting prediction models for the effort required to understand a component.

Hypothesis ($\mathbf{H_2}$): Prediction models for the effort required to understand a component created using just the participants' experience variables as predictors provide better prediction than using the median as an estimate.

Hypothesis ($\mathbf{H_3}$): Combining both component related metrics and the participants' experience variables leads to a significantly increased efficiency of the obtained prediction models in comparison with the prediction models obtained in our previous work [24, 23].

Hypothesis ($\mathbf{H_4}$): Combining both component related metrics and the participants' experience variables leads to a significantly increased efficiency of the obtained prediction models in comparison with the prediction models that use just the participants' experience variables.

Hypothesis ($\mathbf{H_5}$): Combining both component related metrics and the participants' experience variables leads to a significantly increased efficiency of the obtained prediction models in comparison with the prediction models that use just the hierarchical understandability metrics.

D. Study design

In this section we describe the design of our study.

1) Subjects: The participants of the study were 49 master students. The study took place within the Advanced Software Engineering (ASE) lecture at the University of Vienna in the Winter Semester 2013.

2) Objects: The object of our study was the Soomla Android store[1] system, Version 2.0. It is an open source cross platform framework that supports virtual economy in mobile games, and encourages better game design and faster development. The reasons why we chose the given system are explained in more details in our previous work (see [24]).

3) Instrumentation: The following artefacts were provided to the participants.

Architectural documentation of Soomla: The architectural documentation that is handed out to the participants includes the description of the conceptual architecture of the systems together with the frameworks and technologies used for the system implementation. For the architectural representation of the system we use a UML component diagram that shows the components and their inter-relationships (connectors). Traceability links that represent the relationships between the components and the low level source code classes are also provided.

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

Source code access: The access to the source code of the system was Browser-based. It supported easily navigating through the components and opening the source code of the realized classes. This was enabled in a Lab environment on prepared computers.

A questionnaire to be filled-in by the participants: The first part of the questionnaire is related to the participants' experience that they had to rate. The second part of the questionnaire contains the questions related to the understandability of the 7 architectural components. In order to correctly answer the questions the participants had to fully understand the functionalities of each component by studying the relationships (as well as the roles of those relationships) that exist between the classes inside a component and the relationships that exist between the classes inside a component and the classes outside of that component. 4 true/false questions were provided to be studied for each component in the architecture. The 7 studied components were randomized so that 7 random combinations of them were generated and randomly assigned to the participants. The randomization enabled us to get more or less balanced data for all the components in terms of equalizing the fatigue effects or the lack of time needed to complete all required tasks.

In order to measure the time that the participants spent on studying each of the components we provided a table with time slots. Each slot contains a start and a stop time. The start time indicates the time when the participants started studying a component while the stop time indicates the time when they finish it. Several slots were provided for each component in case that the participants wanted to study a component several times. The format used for writing the time is *hour : minute*.

The time limit for the whole study was 90 minutes. The above explained instruments are contained in a document that can be found on the Web[2].

E. Execution

1) Data collection: According to the experience of the participants most of them have more than 3 years of programming experience. Many of the participants also have industrial programming experience while only a few of them have Android and game programming experience.

The descriptive statistics (mean, median, and standard deviation) related to the time and the percentage of the correct answers variables is shown in Figure 1. From our results we excluded the participants that have less than one year of programming experience, the participants who did not specify both start and stop time for the studied components, and some participants who spent only a very short time in studying the components (see [24] for more details).

The data related to the metrics we aim to explore is shown in Table IV. The metrics are independently calculated by two architects who studied the system in order to avoid misinterpretations in their calculations.

If we take a look at Figure 1 we can say that the obtained time for the first three components (*C1*, *C2* and *C3*) is significantly lower than the time for the remaining four components. This observation is logical and expected since the first three components contain a smaller number of classes in

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

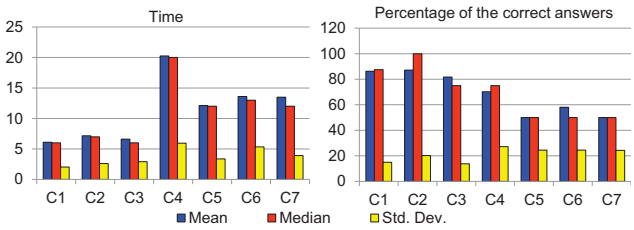

Figure 1. Descriptive statistics for the time and answers' correctness variables (from [23])

comparison to the other four. Another observation is related to Component *C4*. The average time needed to study this component is significantly higher than the time needed to study the Components *C5*, *C6* and *C7*. Consequentially the percentage of the correct answers for the Components *C5*, *C6* and *C7* is decreased in comparison to the Component *C4*. Even though it seems expected that the percentage of the correct answers decreases for the components that have many classes simply because of the higher amount of information that needs to be studied, which in turn increases the probability of missing some relevant information, it seems also that the participants spent a bit less time for studying the Components *C5*, *C6* and *C7* than necessary (or at least for the Component *C7* which has the same number of classes as the Component *C4*) in order to score better and achieve the higher percentage of the correct answers. With respect to this and the discussion in Section III-B the percentage of the correct answers variable is used to help in estimating the time required to fully study a component and achieve the maximal correctness of 100%.

Component level metrics	MSC	NAC	DMC	NDC	CRW	DMH
Security (C1)	2	1	2	0	1	1
CryptDecrypt (C2)	5	5	3	0	1.8	2
PriceModel (C3)	3	1	2	0	1.25	3
GooglePlayBilling (C4)	11	4	2	0	3.5	1
StoreController (C5)	8	2	4	0	1.33	1.87
DatabaseServices (C6)	8	5	4	0	2.29	1.87
StoreAssets (C7)	11	6	3	0	1.43	2.64

Table IV
HIERARCHICAL UNDERSTANDABILITY METRICS VALUES

2) Validation: To prevent the participants from using forbidden materials and talking to each other at least one observer was present in the lab during the study execution. It also enabled the participants to pose clarification questions. The materials given to the participants are collected before any of them left the lab. There were no cases where the participants behaved unexpectedly.

IV. ANALYSIS

The following statistical tests and parameters are used in our study for analysing the data:

- The Variance Inflation Factor (VIF) and the Condition Number (CN) calculation – Collinearity Analysis
- Multivariate Regression Analysis
- Second-order corrected Akaike Information Criterion (AICc)

The above mentioned analyses are performed using the programming language R [21].

A. Collinearity Analysis

Collinearity analysis aims at indicating the variables that are highly correlated with each other and should be excluded from the set of all possible predictors. In order to test for the possible correlations we calculate the Condition Number (CN) and the Variance Inflation Factor (VIF). VIF values greater than 10 and CN values greater than 30 suggest high correlation [5].

By calculating the VIF and CN values we find that there are no multicollinearity problems in the set of hierarchical understandability metrics as well as in the participants' experience variables. As we discussed above we would like to examine a model where all the studied variables (including the metrics from our previous work) are taken into account, i.e. the hierarchical understandability metrics, the participants' experience, the package-level metrics studied in [24] and the graph-based metrics studied in [23]. Combining all these sets together introduces multicollinearity problems since there are metrics in multiple sets that measure the same concepts (size, coupling, and cohesion). After examining the VIF and CN values all graph-based and package-level metrics can be excluded from the model.

B. Multivariate Regression Analysis

In this part of the analysis we created multivariate regression models that can be used for predicting the time variable. To prevent the over-fitting of the data, i.e. to enable more efficient generalization of the results we perform Mallows' Cp calculation for creating the prediction models [17].

To check the accuracy of the obtained prediction models we calculated a goodness of fit measure using the following equation based on the absolute deviation of the median (assuming X_i is the prediction and Y_i is the actual value):

$$A(accuracy) = \frac{\sum_i |Y_i - X_i|}{\sum_i |Y_i - median(Y_i)|}$$

The A value greater than 1 means that there is no evidence that the prediction is better than using the median as an estimate. The value (1-A) is a robust analogue of R^2, so the following guidelines based on those proposed by [15] can be used for the effect size calculation: the (1-A) values in the range of 0 to 0.0372 represent a small effect size, the values in the range of 0.0372 to 0.208 represent a medium effect size while the values in the range of 0.208 to 0.753 represent a large effect size. Furthermore, for good prediction models the residuals have to be normally distributed which is the case with our data. We further calculate the coefficient of determination (R^2) for the obtained models that describe how well the model fits a set of data. The significance of this coefficient is tested using the F test [9].

Another useful technique for overcoming the over-fitting problem is cross-validation analysis [12]. We applied the 10-fold cross-validation technique on our data [1]. The results of the cross-validation analysis corroborate the results of the Mallows' Cp analysis and confirm their validity.

To test Hypothesis $\mathbf{H_1}$ we created prediction models for the hierarchical understandability metrics. There are in total 5 models that satisfy Mallows' Cp criterion. The obtained models have the adjusted R^2 values around 91 %. The calculated models' effect size is around 39 % which represents a large effect size. Therefore with respect to the obtained results, we can say that the Hypothesis $\mathbf{H_1}$ of our study is supported.

To test Hypothesis $\mathbf{H_2}$ we created prediction models that use the experience of the participants as predictors. In this case we obtain several models that satisfy Mallows' Cp criterion. Comparing to the other obtained models we can say that these models are much less accurate and efficient, i.e. the accuracy measures are much higher (96-100 %) and the adjusted R^2 measures are much lower (around 61 %). However, the accuracy of some of the obtained models is lower than 1 which means that the prediction is better than using the median as an estimate. Therefore it has been demonstrated that the Hypothesis $\mathbf{H_2}$ of our study is supported.

C. Models Comparisons

To test the hypotheses $\mathbf{H_3}$, $\mathbf{H_4}$, and $\mathbf{H_5}$ we create prediction models that combine the model related metrics and the participants' experience. According to the discussion in Section IV-A, the graph-based and the package-level metrics can be excluded from the model because of the indicated multicollinearity problems.

To compare the efficiency of different prediction models, we calculate the difference between the AICc (second-order corrected Akaike Information Criterion) values (ΔAICc) for those models [6]. If the obtained difference (ΔAICc) is lower than 4 we can say that there is no significant difference in the prediction capabilities (power) of the given two models [6]. If the difference is in the range [4,7], we can say that there is a significant difference in the prediction capabilities, and, if the difference is greater than 10, a very strong difference exists [6]. We compare the best models from each group in terms of the AICc measure.

From the obtained results, the ΔAICc measure is greater then 10 between the model that considers the hierarchical metrics together with the participants' experience and the models that consider the participants' experience, the graph-based metrics, and the package-level metrics separately. The models considering graph-based and package-level metrics are taken from our previous studies [23, 24]. Regarding the difference between the model that considers the hierarchical metrics together with the participants' experience and the model that just includes the hierarchical understandability metrics no significant difference exists (ΔAICc 1=-1.88). Based on the obtained results we can say that the Hypotheses $\mathbf{H_3}$ and $\mathbf{H_4}$ of our study are supported while the Hypothesis $\mathbf{H_5}$ is not supported.

V. Validity evaluation

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

Conclusion validity: The conclusion validity indicates to which extent the conclusions of a study are statistically valid. While the number of participants in our study is quite fair the dataset consisting of 7 components is limited to a relatively small-size dataset due to the limited time of the study session. We plan to increase the number of studied components in our future work.

Construct validity: The construct validity describes the degree to which the variables used in a study are accurately measured by the applied instruments. A possible threat might be related to the time variable. To minimize that threat we put a reminder before the text related to each component to remind the participants to write the time appropriately. The component level metrics are calculated automatically with the help of the tool ObjectAid UML Explorer[3].

External validity: The external validity is related to the degree to which the results of the study can be generalized to a broader population. There might be some classes in the system that are much bigger than other classes. In that case the number of classes in a component will not appropriately capture the component size (in our case there are no big deviations in size, however). Please note also that this might also be considered as inappropriate design, i.e. very big classes can be divided into smaller classes that consist of one or a set of closely related functionalities. Anyway the given observation can be further examined in order to see how the deviations in the size of classes affect the obtained results.

Regarding our subjects we already showed that they have substantial experience including industrial background. However in order to generalize our findings it is necessary to carry out more studies with professionals.

VI. Conclusions and future work

In this article we studied the understandability of architectural components using the hierarchical understandability metrics introduced in the work by Hwa et al. [14], the personal factors of participants like experience and expertise, and the combinations of both personal factors and component level metrics (the previously studied and the newly introduced). The obtained results are compared to the results obtained in our previous studies with the aim of improving the efficiency of the previously obtained prediction models for the understandability effort. On the one hand, the prediction models that consider the newly introduced hierarchical understandability metrics are significantly better in predicting the understandability effort than the models obtained in our previous studies or the models that include just the participants' experience variables. On the other hand, those models are not significantly different in the prediction from the models that combine both model related metrics and the participants' experience variables. This means that from all studied models it is enough to consider them for the prediction. This result is from our point of view intuitive, as those metrics are originally designed to assess the understandability of the modular design of a system. The participants' experience is also important and can predict a significant amount of variance in the data but the obtained models are not as accurate as the models that use the component level metrics, i.e., the metrics related to the software model itself.

Acknowledgement

This work was supported by the Austrian Science Fund (FWF), Project: P24345-N23.

[3]www.objectaid.com

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] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.
[5] 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.
[6] K. Burnham and D. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, 2002.
[7] S. Chidamber and C. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, Jun 1994.
[8] 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.
[9] P. Dalgaard. *Introductory Statistics with R*. Springer, Jan. 2004.
[10] M. O. Elish. Exploring the relationships between design metrics and package understandability: A case study. In *ICPC*, pages 144–147. IEEE Computer Society, 2010.
[11] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 2nd edition, 1998.
[12] A. Field, J. Miles, and Z. Field. *Discovering Statistics Using R*. SAGE Publications, 2012.
[13] V. Gupta and J. K. Chhabra. Package coupling measurement in object-oriented software. *J. Comput. Sci. Technol.*, 24(2):273–283, Mar. 2009.
[14] J. Hwa, S. Lee, and Y.-R. Kwon. Hierarchical understandability assessment model for large-scale oo system. In *Software Engineering Conference, 2009. APSEC '09. Asia-Pacific*, pages 11–18, Dec 2009.
[15] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. K. Sjøberg. Systematic review: A systematic review of effect size in software engineering experiments. *Inf. Softw. Technol.*, 49(11-12):1073–1086, Nov. 2007.
[16] 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.
[17] M. Kobayashi and S. Sakata. Mallows' cp criterion and unbiasedness of model selection. *Journal of Econometrics*, (3):385–395.
[18] F. Losavio, L. Chirinos, N. Lvy, and A. Ramdane-Cherif. Quality characteristics for software architecture. *Journal of Object Technology*, 2(2):133–150, 2003.
[19] R. C. Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.
[20] T. J. McCabe. A complexity measure. *IEEE Trans. Softw. Eng.*, 2(4), July.
[21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.
[22] H. Reijers and J. Mendling. A study into the factors that influence the understandability of business process models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(3):449–462, May 2011.
[23] S. Stevanetic and U. Zdun. Exploring the relationships between the understandability of architectural components and graph-based component level metrics. In *Proceedings of the 14th International Conference on Software Quality (QSIC)*, QSIC 2014, Dallas, USA, 2014. IEEE Computer Society.
[24] 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.
[25] S. Stevanetic and U. Zdun. Software metrics for measuring the understandability of architectural structures: a systematic mapping study. In J. Lv, H. J. Zhang, and M. A. Babar, editors, *EASE*, pages 21:1–21:14. ACM, 2015.
[26] C. Wohlin. *Experimentation in Software Engineering: An Introduction: An Introduction*. The Kluwer International Series in Software Engineering. Kluwer Academic, 2000.