Empirical Evaluation of the Understandability of Architectural Component Diagrams

Srdjan Stevanetic, Muhammad Atif Javed and Uwe Zdun
Software Architecture Research Group
University of Vienna, Austria
srdjan.stevanetic|muhammad.atif.javed|uwe.zdun@univie.ac.at

ABSTRACT

The architecture of a software system plays a crucial role during evolution and maintenance, as it provides the means to cope with the inherent system complexity by abstracting from implementation and design details. Architectural component models represent high level designs and are frequently used as a central view of architectural descriptions of software systems. Hence, understandability of those models is crucial as they play a key role in supporting the architectural understanding of a software system. In this paper we present the results from a study we carried out to examine to which extent the software architecture could be conveyed through architectural component diagrams. The statistical evaluation of the results shows that metrics such as the number of components, number of connectors, number of elements, and number of symbols used in the diagrams can significantly decrease architectural understandability when they are above and below a certain, roughly predicted threshold. Also, our results indicate that architectural understandability is linearly correlated with the perceived precision and general understandability of the diagrams.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.8 [**Software Engineering**]: Metrics

Keywords

Software Architectures, Architectural Component Models, Software Metrics, Empirical Evaluation.

1. INTRODUCTION

The architecture of a software system plays a crucial role in the system's lifecycle since it guides its evolution and maintenance. That is, the software architecture is a means to cope with the inherent system complexity and to coordinate different maintenance tasks (e.g., assigning the right people to the right problems or planning and monitoring the activities). In order to successfully evolve large-scale systems it is essential to understand their architecture before delving into details of the implementation, which is a time and effort consuming activity. In particular, large amounts of data (e.g., several millions lines of code) have to be analysed and abstracted to an architectural level and important information is often hidden among irrelevant data.

How complex or simple the structure of the system is depends essentially on the way we describe it. Architectural component and connector models (or component models for short) are frequently used as a central view of the architectural descriptions of software systems. Component models used in software architecture represent high-level abstractions of the software system and concern only the critical design decisions about the software system [8].

Understandability is a critical aspect of architectural models because of their focus on abstraction and conveying the "big picture" of the system. However, 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 understandability (see e.g. [6, 4]). To the best of our knowledge, there is no existing empirical study on the understandability of architectural component diagrams (the two previously mentioned studies [6, 4] examined understandability at the package level).

In this paper we present the results of a study we carried out to examine to which extent the software architecture can be conveyed through architectural component diagrams. In particular, we conducted an empirical study based on a survey with participants of the SHARK 2012 workshop[1] (33 participants) and an identical replication in our Advanced Software Engineering master course (35 participants). In the study the participants were asked to study and rate 16 architectural component diagrams from existing software systems. We explore if it is possible to evaluate the subjects' ratings of the understandability of the given architectures using some simple size metrics (below referred to as variables) we collected from the component diagrams.

This paper is organized as follows: In Section 2, we briefly discuss the related work. In Section 3 we describe the study design. Section 4 describes the statistical methods we ap-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WICSA '14, April 07 - 11 2014, Sydney, NSW, Australia
Copyright 2014 ACM 978-1-4503-2523-3/14/04 ...$15.00.

[1]Seventh Workshop on SHAring and Reusing architectural Knowledge (SHARK 2012). Hosted at Joint 10th Working Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA/ECSA 2012), Helsinki, Finland, August 20, 2012.

plied and the analysis of our data. In Section 5 we discuss the threats to validity of the study. In Section 6 we conclude and discuss future directions of our research.

2. RELATED WORK

As we mentioned before, so far in the software architecture literature we find only a very few studies that provide empirical evidence regarding architectural understandability. In particular, one existing study examines the influence of package coupling on the understandability of the software systems [6], while another one examines the relationships between some package-level metrics and package understandability [4].

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 [13]. 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 [13]. In our study we use the model user rating of model understandability.

Even though so far there are no rigorous empirical studies of architectural component model understandability, aspects like fault density and reuse of components have been studied before. Fenton and Ohlsson have studied the relations of fault density and component size [5]. Mohagheghi et al. have studied the comparison between software reuse and defect density and stability [12]. Their study is based on the historical data on defects, modification rate, and software size. Malaiya and Denton identify the component partitioning and implementation as influencing factors to determine the "optimal" component size with regard to fault density [11].

A number of authors proposed ways to improve the understandability of architectural models through additional models or documentation artefacts. A major research direction deals with documenting architectural decisions and architectural knowledge in addition to component models [2, 8, 17]. Another major research direction deals with architectural views [3, 7, 10] which enable different stakeholders to view the architectures from different perspectives. Both research directions only complement component models with additional knowledge, but neither of them can fully resolve understandability issues related to the component models themselves.

3. EMPIRICAL STUDY DESCRIPTION

For the study design we have followed the experimental process guidelines proposed by Kitchenham et al. [9] and Wohlin et al. [16]. 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.

3.1 Goals, variables, and hypotheses

The main idea of this study is to explore to which extent the software systems architecture could be conveyed through architectural component diagrams. In order to better explain the hypothesis and the main goal of the study we will firstly explain the variables that we used. We can differentiate two groups of variables. The first group of variables was collected from the participants of the study, and it represents the subjects' rating variables, while the second group of variables was collected from the component diagrams. The first group of variables includes five independent variables: demographic information (programming experience, software design experience, professional experience, and affiliation) and the subjects' expertise for the application domain of the diagrams. It also includes the three dependent variables: precision/accuracy, general understandability, and architectural understandability of the diagrams:

- The precision/accuracy variable measures in how far the given diagram is precise compared to other potential illustrations of the same system. For instance, is it necessary to add some elements (entities) in the diagram to increase the precision with respect to the system description, are the names of the architectural entities coherent and informative (for instance like LoadBalancer and unlike ServerProcess which might be too plain), are the links between components informative (e.g., giving clear interface/connector descriptions)? Also, very abstract model representations with few elements might not carry enough information about the system.

- The general understandability variable measures the readability of the diagrams and reasonable use of graphical elements. For instance, is the meaning of text and names recognizable and clear to the reader, are the elements and their connections placed according to a reasonable law (random placement distract readers), are the symbols used in the diagram understandable and clear, is the diagram too complex (e.g. because of a large number of elements and symbols that will cause problems with regard to human perception limits)? All mentioned aspects affect building a mental model of the architecture of the given software system. This process is based on the information present in the diagram and information from already seen diagrams and previous experiences (prior knowledge).

- The architectural understandability variable measures in how far the given diagram supports architectural ("big picture") understanding of the system. For example, it is related to questions like is the diagram sufficient to describe all relevant architectural concepts of the system gained from the system description and the reader's previous knowledge, or are the system's major services appropriately mapped to the components working in the collaboration to deliver those services? When the system consists of some architectural patterns, are they reflected (e.g. conveyed) in an appropriate way in the diagram?

As this is the initial study among a series of studies we plan to conduct in establishing empirically grounded guidelines supporting the design of architectural component diagrams the idea of this study is to try to evaluate some general concepts of the software architecture representation that users face with in the process of diagram comprehension. On the one hand, the user is faced with a visual representation of the diagram trying to build a mental model of the system. Here, the appropriate use of graphical elements

and diagram readability plays an important role. The semantics of the elements and the text in the diagram are captured through the users previous knowledge and familiarity with a domain semantics of the system. These factors are captured by the general understandability variable in our study. On the other hand, redundant information as well as not enough information makes the diagrams imprecise (i.e., relevant information remains hidden). This might hamper the comprehension of the diagram. Thus, the precision of the diagrams, captured by the level of abstraction and the amount of information contained in the diagrams, also plays an important role and is captured by the precision variable in our study. Both variables contribute to the architectural understandability that aims to capture the "big picture" of the system referring to all architecturally relevant aspects. The explained understandability concept used in our study can be coarsely mapped to the understandability of the software architecture explained in the work by Anderson et al. [1]. The authors indicated two important views for the software architecture understanding: semantic view and cognitive view. In the semantic view the importance of diagram notations that carry the semantics about the system's application domain is emphasised and it is captured by the general understandability variable in our study. In the cognitive view the appropriate matching between the diagram representations and the solving tasks is emphasized and it is captured by the precision variable in our study.

To illustrate the relations between these three variables, consider a diagram with only a few elements which represents a very high level of abstraction of the system implementation. This diagram can be understandable in general (simple visual representation, clear elements' semantics) but imprecise because of the lack of information in the diagram and its inability to model all relevant details of the system. On the other hand, the diagram might also be very precise (informative names and appropriate abstraction level that capture the system key concerns well) but less understandable in general because of its confusing visual representation and the participant's lack of knowledge about the diagram's domain. Both, the lack of precision in the first example and the problems regarding general understandability in the second example, will likely deteriorate architectural understandability of the diagrams.

All these dependent variables have been explained to the participants before the study execution in order to avoid possible confusions. During the discussion phase of the study (see Section 3.3) further qualitative data was gathered from group discussions among the participants to confirm and help understanding the participants' answers to the study questions related to the dependent variables (see Section 3.2.3).

The second group of variables includes four size variables. They include 3 independent variables number of components ($NCOMP$), number of connections between the components ($NCONN$), and number of symbols ($NSYM$) in the diagram, and the fourth variable, total number of elements, summing up number of components and number of connections ($NELEM$). For the $NCONN$ variable, regardless whether the connection between the components is one-way, two-way, or without a specific direction, we counted either case as one connection, assuming that the difference between a one-way or two-way connections does not have a big influence on the architectural understandability. The $NSYM$ variable is calculated by taking into account different symbols that can affect the architectural understandability like different component symbols (e.g., boxes for components or special symbols such as databases, clients, protocols), symbols for the connections between the components (e.g., different kinds of lines), symbols for interfaces (e.g., required, provided interfaces), UML stereotypes, and so on.

This second group of variables is used to evaluate the three dependent variables (precision/accuracy, general understandability, and architectural understandability). Regarding general understandability we expect that high values of the size variables lead to lower general understandability because of the higher cognitive load of large models and the human perception limits. Regarding precision, those higher values can lead to less precise diagrams because of redundant information presented in the diagrams that makes them more imprecise. Generally, it is not feasible to capture everything we want to model in a single model. This model would be too big and confusing with no clearly delimited concerns. Diagrams with high/very high numbers of entities usually do not focus on particular issues or concerns and/or they mix different issues or concerns. For example, one diagram might clarify how to decompose and implement the system (for programmers), and another one might concentrate on external services that a platform provides or on a view for non-technical stakeholders. However, mixing all three aspects in one diagram is expected to have a negative effect on our dependent variables. For low levels of the size variables we expect that they also affect imprecise diagrams because of their high abstraction level and inability to model all relevant details of the system. Having these two facts in mind we expect that middle values of the size variables will lead to higher levels of architectural understandability than low or high values.

The dependent variables together with their scale types, units, and ranges are shown in Figure 1. All these variables are measured using a ten point Likert-scale [15]. The independent variables are shown in Figure 2.

Description	Scale type	Range
Precision of the diagrams	Interval	Ten point Likert-scale: 1-the lowest, 10-the highest
General understandability of the diagrams	Interval	Ten point Likert-scale: 1-the lowest, 10-the highest
Architectural understandability of the diagrams	Interval	Ten point Likert-scale: 1-the lowest, 10-the highest

Figure 1: Dependent variables

Description	Scale type	Unit	Range
Programming experience	Ordinal	Years	4 categories: 0,1-3,3-7, >7
Software design experience	Ordinal	Years	4 categories: 0,1-3,3-7, >7
Professional experience	Ordinal	Years	4 categories: 0,1-3,3-7, >7
Affiliation	Nominal	-	Categories: academia, industry, other
Expertise for the diagram's application domain	Interval	-	Ten point Likert-scale: 1- the lowest, 10 - the highest
Number of components	Ratio	Component	Positive natural numbers including 0
Number of connectors	Ratio	Connector	Positive natural numbers including 0
Number of symbols	Ratio	Symbol	Positive natural numbers including 0

Figure 2: Independent variables

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

$\mathbf{H_{01}}$: There is a significant positive correlation between the general understandability and precision of the diagrams on one side and the architectural understandability on the other side.

H_{02}: There is a significant negative correlation between one or more of the size variables $NCOMP$, $NCONN$, $NELEM$, and $NSYM$ and the general understandability variable.

H_{03}: There is a significant positive correlation between the subject's expertise for the diagrams application domain and the general understandability of the diagrams.

H_{04}: Middle values of the size variables ($NCOMP$, $NCONN$, $NELEM$, and $NSYM$) significantly increase the architectural understandability compared to low or high values.

3.2 Study design

To test the hypotheses, we conducted two executions of the study using exactly the same design. The first execution took place at the 7th Workshop on SHAring and Reusing Architectural Knowledge (SHARK 2012). The second execution took place as part of the Advanced Software Engineering (ASE) master lecture at the University of Vienna in the Winter Semester 2012.

3.2.1 Subjects

The subjects of the study are the 33 attendees of the SHARK 2012 workshop and the 35 master students of the ASE lecture.

3.2.2 Objects

The objects of the study are 16 architectural component diagrams that vary with respect to 3 factors: size and level of detail (number of components, connectors, modelling symbols used), diagrams' application domain (to capture the domain knowledge of the participants we used diagrams of different application domains), and diagrams' representation (UML diagrams versus informal box-and-line diagrams).

Please note that the domain of the diagrams is more or less known to the participants where some of them have more, some less experience with the systems (see the participants expertise for diagrams' application domain shown in Figure 5).

3.2.3 Instrumentation

Each subject of our study received one questionnaire in which a number of variables were measured through the subject's ratings. All ratings were given on paper and no other materials or computer access were provided. On the first page of the questionnaire, they had to rate their experience, i.e. the independent variables programming experience, software design experience, professional experience, and affiliation. On the following pages of the questionnaire, the 16 component diagrams described in Section 3.2.2 were shown, each together with four questions about the independent variable expertise for the diagram's application domain and the dependent variables:

- How do you rate the expertise for the application domain of the system documented here?
- How understandable is the component diagram in general?
- How accurate/precise is the component diagram compared to other potential illustrations of the same system?
- In how far does this component diagram support architectural (i.e., "big picture") understanding?

The size variables $NCOMP$, $NCONN$, and $NELEM$ are calculated automatically through a tool that we have developed, while $NSYM$ variable is obtained manually.

3.3 Execution

3.3.1 Preparation

The variables we collected from the diagrams (as explained before) are shown in Figure 3. The tasks for the participants consisted of three steps. The total time limit for the whole study was 2 hours.

- *Individual completion of the questionnaire.* No time limit was given for this task. All participants concluded the task in a time between 40 and 60 minutes.
- *Group discussion (5-10 people in each group).* The group discussion started when all members of the group were ready with Task 1. The group discussion time was for all groups between 40 and 60 minutes.
- *Reporting results in front of the other groups.* Each group reported their discussion results in a 5-10 minutes presentation.

Metrics	Diagrams															
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
NCOMP	13	23	9	4	14	146	5	5	11	10	9	16	22	15	42	9
NCONN	12	16	9	3	4	0	5	5	20	12	11	10	18	13	17	3
NSYM	3	4	4	4	6	1	7	5	3	3	6	10	7	4	11	6
NELEM	25	39	18	7	18	146	10	10	31	22	20	26	40	28	59	12

Figure 3: Variables obtained from the component diagrams

During the group discussion the groups discussed about different topics regarding categories of the diagrams, how to improve the architectural understandability of the diagrams and so on. Those results will not be presented in this paper due to space reasons.

3.3.2 Data collection

The data collection was performed as planned in the design. At least one observer was present in the room during the whole study execution time to assure that participants did not use forbidden materials and did not talk to each other. After execution, all materials were collected before any of the participants left the room. There were no situations in which participants behaved unexpectedly.

The relevant data regarding the participants' demographic information and experience are shown in Figure 4. According to the experience of the participants we can say that the participants of the SHARK workshop have substantially more programming and architecture experience than the participants of the ASE lecture, but the students also have a significant level of experience. (A large number of our master students have a number of years of part-time or even full-time working experience in industry, which is confirmed by these numbers.)

Based on the questions for the diagrams we collected four subject rating variables: expertise for the application domain of the diagrams, general understandability of the diagrams, accuracy/precision of the diagrams, and architectural understandability of the diagrams. Answering

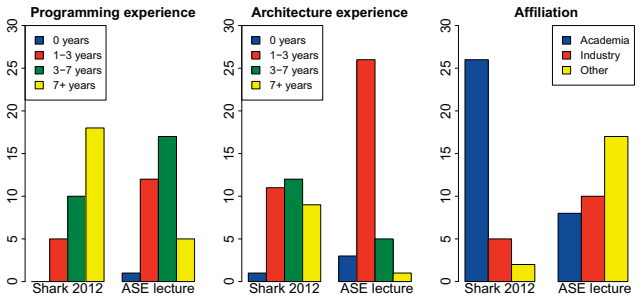

Figure 4: Experience and affiliation of the participants (Shark 2012 and ASE lecture)

the questions was based on a 10 point Likert scale in which "1" corresponds to "imprecise/not understandable" diagrams while "10" corresponds to "very precise/easily understandable" diagrams. Figure 5 shows the medians of those variables (*medianExp*, *medianPrec*, *medianUndGen* and *medianUndArch*) for both groups of participants.

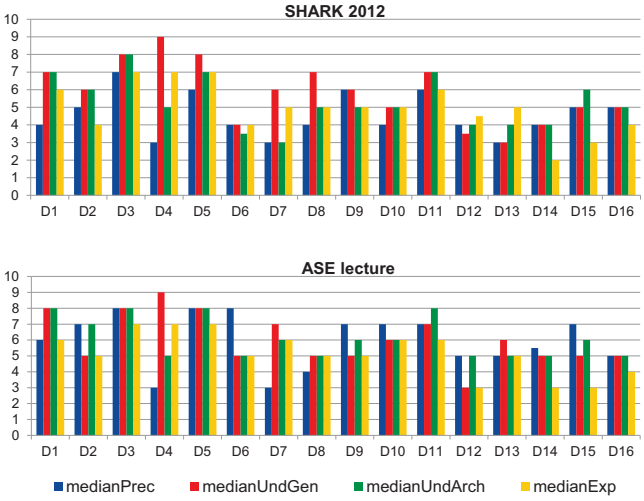

Figure 5: Medians of the subjects' rating variables

3.3.3 *Validation*

We collected all the answers from participants, checking if they were complete. As all of them were complete and the participants had at least medium experience in programming and software architecture/design, we consider their evaluation reliable.

4. ANALYSIS

Because the data obtained from the questionnaire are ordinal or rank variables we applied the following, non-parametric statistical tests.

- The Spearman rank correlation test for testing the correlation between two variables
- The Kruskal–Wallis test for comparison of a location shift between more than two variables

There is not much to be gained from formally combining the results because both executions produced significant results. Also, simply combining the data into one big analysis does not make much sense because there is a significant difference in medians between some of the observed variables of the different groups. For statistical analysis of the obtained data, we used the programming language R [14].

4.1 Testing Hypotheses

4.1.1 *Testing the Correlation Between the Variables*

In order to test the first 3 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. Figure 6 shows the Spearman's coefficients and the corresponding p-values for both study executions (I: Shark 2012, II: ASE course).

	NCOMP	NCONN	NELEM	NSYM	Study
medianPrec	rho=0.082, p=0.763	rho=0.126, p=0.642	rho=0.089, p=0.742	rho=-0.070, p=0.796	I
	rho=0.482, p=0.059	rho=0.107, p=0.6934	rho=0.469, p=0.067	rho=-0.366, p=0.163	II
medianUndGen	**rho=-0.628, p=0.009**	rho=-0.320, p=0.227	**rho=-0.656, p=0.005**	rho=-0.197, p=0.464	I
	rho=-0.49 , p=0.054	rho=-0.241, p=0.368	**rho=-0.507, p=0.045**	rho=-0.145, p=0.591	II
medianUndArch	rho=-0.121, p=0.656	rho=0.117, p=0.669	rho=-0.152, p=0.554	rho=-0.085, p= 0.755	I
	rho=-0.052, p=0.847	rho=0.204, p=0.448	rho=-0.057, p=0.834	rho=-0.108, p=0.692	II

	medianExp	medianPrec	medianUndGen	medianUndArch	Study
medianExp	--	rho=0.1270, p= 0.6394	**rho=0.7700, p= 0.0005**	rho=0.4977, p= 0.0498	I
	--	rho=0.1848, p= 0.4931	**rho=0.9121, p= 8.62e-07**	rho=0.5668, p= 0.0220	II
medianPrec	rho=0.1270, p= 0.6394	--	rho=0.3373, p=0.2014	**rho=0.7197, p=0.002**	I
	rho=0.1848, p= 0.4931	--	rho=0.0496, p=0.8551	**rho=0.5749, p=0.0198**	II
medianUndGen	rho=0.7700, p= 0.0005	rho=0.337, p=0.201	--	**rho=0.6728, p=0.0043**	I
	rho=0.9121, p= 8.6e-07	rho=0.049, p=0.8551	--	**rho=0.5246, p=0.037**	II
medianUndArch	rho=0.4977, p= 0.0498	rho=0.7197, p=0.002	rho=0.6728, p=0.0043	--	I
	rho=0.5668, p= 0.0220	rho=0.5749, p=0.0198	rho=0.5246, p=0.037	--	II

Figure 6: The Spearman correlation coefficients and corresponding p-values

Regarding the hypothesis $\mathbf{H_{01}}$, there is a significant positive correlation between the precision and the general understandability on one side and the architectural understandability of the diagrams on the other side, for both study executions. Regarding the hypothesis $\mathbf{H_{02}}$, the number of elements in the diagram has significant negative correlation with the general understandability (for both study executions) as well as the number of components in the diagram (for the first study execution while for the second execution the value is just below the threshold). The number of symbols and the number of connections in the diagrams also have negative correlation with the general understandability but it is not significant.

Actually, the total number of elements in the diagram is the most convenient indicator in terms of the affected cognitive load in the human perception. For our set of diagrams the total number of elements is mostly determined by the number of components in the diagrams (that is the reason why *NCOMP* variable also shows significant correlation with the general understandability), while the number of links varies significantly. For instance, for the diagram D6 we have 146 components and 0 links, for the diagram D5 we have 14 components and 4 links, for the diagram D15 we have 42 components and 17 links. Regarding the number of

symbols, the most numbers lie in the range [3, 6] so it hardly affects any cognitive load.

Regarding the hypothesis $\mathbf{H_{03}}$, there is a significant correlation between the subjects' expertise for the diagrams application domain and the general understandability of the diagrams for both study executions (see Figure 6) which proves that domain knowledge plays an important role in the diagrams comprehension.

Based on the results of the Spearman rank correlation test we accept the hypotheses $\mathbf{H_{01}}$, $\mathbf{H_{02}}$, and $\mathbf{H_{03}}$ of our study.

4.1.2 Comparison of a Location Shift Between More Than Two Variables

To test the hypothesis $\mathbf{H_{04}}$ we applied the Kruskal-Wallis test. The first question is which values of the size variables are low, middle and high. Some intuition might be that more than 20 or 30 elements in the diagram can affect human perception limits and high cognitive load as well as imprecision in terms of mixing different issues with the main concepts of the system. We first observed the distribution of the architectural understandability variable with respect to $NCOMP$, $NCONN$, $NELEM$, and $NSYM$ variables obtained from the component diagrams in order to roughly find possible levels (groups) of those variables that can effect a significant difference in the distribution of architectural understandability variable with respect to the discussion in Section 3.1. After roughly distinguishing those groups we applied the Kruskal-Wallis test and the corresponding post-hoc tests in order to find if there is a significant difference between some of those groups. As a post-hoc test we used the Mann-Whitney test with Holm correction. We distinguished 3 groups for the $NCOMP$, $NCONN$ and $NELEM$ variables while for the $NSYM$ variable we distinguished 2 groups (in this case we applied Wilcoxon rank-sum test). The resulting p-values obtained after applying the post-hoc tests for both executions of the study for the $NCOMP$, $NCONN$ and $NELEM$ variables are shown in Figure 7.

NCOMP groups	**1 (≤5)**	**2 (>5 & <15)**	**3 (≥15)**	**Study**
1 (≤5)	–	–	–	I II
2 (>5 & <15)	**3.1e-05** **2.9e-05**	–	–	I II
3 (≥15)	0.20 0.85	**6.0e-06** **8.4e-08**	–	I II
NCONN groups	**1 (≤3)**	**2 (>3 & ≤17)**	**3 (>17)**	**Study**
1 (≤3)	–	–	–	I II
2 (>3 & ≤17)	**0.020** **0.002**	–	–	I II
3 (>17)	0.640 0.604	**0.007** **0.014**	–	I II
NELEM groups	**1 (≤11)**	**2 (>11 & ≤25)**	**3 (>25)**	**Study**
1 (≤11)	–	–	–	I II
2 (>11 & ≤25)	**8.8e-06** **2.3e-06**	–	–	I II
3 (>25)	0.16 0.81	**2.8e-07** **5.3e-10**	–	I II

Figure 7: Kruskal-Wallis post-hoc test p-values with respect to the architectural understandability

Regarding the $NSYM$ variable we distinguished two groups ($NSYM \leq 6$ and $NSYM > 6$). The Wilcoxon rank sum test shows a significant difference for the architectural understandability variable for both executions of the study with the p-values 0.001 for SHARK 2012 and 0.033 for the ASE lecture.

The selected levels (groups) of each variable shown in Figure 7 and for the $NSYM$ variable are the levels for which we obtained the most significant difference in a location shift of the architectural understandability variable. If we increase/decrease the thresholds we obtain less significant differences in the distribution. These thresholds between low, middle and high values should be considered as rough results and more empirical evaluation is needed to precisely determine them and to create guidelines based on them.

By drawing boxplot diagrams for the selected levels (groups) we can see that the architectural understandability variable significantly decreases for low/very low and high/very high values of the $NCOMP$, $NCONN$, and $NELEM$ variables (1st and 3rd group) as well as for high values of $NSYM$ ($NSYM > 6$). As we discussed in Section 3.1 low/very low values of those variables are not sufficient to precisely model all architectural concepts of the system, while high/very high values can lead to a higher cognitive load as well as potential mixing of different concerns in the diagram and therefore deteriorate the architectural understandability. This can also be confirmed by looking at the distribution of the general understandability and precision variables. The general understandability variable significantly decreases for high values of all four variables (for both study executions) leading to higher cognitive load. The precision variable also significantly decreases for low/very low and high/very high values of the size variables (for both study executions).

If we take a closer look at the diagrams, we can observe that the diagrams with low number of elements (D4, D7, D8) are very abstract and as a result imprecise. Also, diagrams with high number of elements (D6, D13, D14) suffer from a lot of visual information and imprecision. For instance, Diagram D6 is very detailed and mixes a lot of views: UI application high level view (UI Framework component), APIs and lower level functionalities such as codecs (Multimedia Framework component), lower level system configuration data and tools (Core component). Diagram D13 which models the Web service middleware shows detailed message content paths in the system through all the layers rather than showing the main interactions between the client, server and Apache Axis processing framework. It also consists two cases of interactions, client- and server-side of Apache Axis as two diagrams which increases the visual complexity of the diagram. Diagram D14 shows a number of main components plus components that deal with lower level interactions like rendering APIs and network context processing. Mixing different concerns in the diagrams was an issue reported from the participants in the discussion part of the study. As shown in Figure 5, all three systems have a low precision and general understandability.

Based on the results of the Kruskal-Wallis test we accept hypothesis $\mathbf{H_{04}}$ of our study.

5. VALIDITY EVALUATION

In this section we discuss the various threats to validity and how we tried to minimize them:

Conclusion validity.

The conclusion validity defines the extent to which the conclusion is statistically valid. While the number of participants in the study is quite fair the number of diagrams can be increased in replications of the study in order to be able

to generalize the results. We plan to add more diagrams as well as to engage more experts who work in industry in our future work.

Another aspect of the conclusion validity might be considered for the hypotheses $\mathbf{H_{02}}$ and $\mathbf{H_{03}}$. Actually, for the hypothesis $\mathbf{H_{02}}$, decreasing of the general understandability variable due to the high number of elements in the diagrams can be also affected by the other 2 factors in the diagrams, diagrams' representation and diagrams' application domain (see Section 3.2.2). In order to reduce the threat to the conclusion validity affected by these two factors we performed the correlation separately for both diagram representations (UML and box-and-line) and in both cases we considered just the diagrams where the subject's expertise for the diagrams' application domain is above the adopted threshold (6 in this case). The correlations were significant for all cases. Regarding the hypothesis $\mathbf{H_{03}}$ we did almost the same procedure. We pursued the correlation separately for both diagram representations and in both cases we identified 2 groups of the diagrams with respect to the number of elements in the diagrams (in order to keep the size factor more-less constant). For the UML diagrams we considered separately the diagrams with [7,12] and [22,31] elements while for the box-and-line diagrams the two groups were [18,26] and [39,146]. In all cases we obtained the significant correlation. Therefore, our conclusion regarding the hypotheses $\mathbf{H_{02}}$ and $\mathbf{H_{03}}$ is considered as valid. By taking into account that the architectural understandability variable is directly correlated with the precision and the general understandability variables, as well as the previously explained threats to the validity of the hypotheses $\mathbf{H_{02}}$ and $\mathbf{H_{03}}$, the threat to the conclusion validity of the hypothesis $\mathbf{H_{04}}$ affected by the other factors in the diagrams is minimized.

Construct validity.

The construct validity is the degree to which the collected variables are accurately measured by their appropriated instruments. Our dependent variables and the expertise for the diagrams' application domain are measured as linguistic levels using subjects' rating. As the subjects involved in this study have at least medium experience in the area of programming and software design/architecture we suppose their ratings could be considered valid. Size variables like $NCOMP$, $NCONN$, and $NELEM$ are collected automatically, using the tool, while $NSYM$ variable is calculated manually for each diagram. There is a low threat to validity that our interpretations for $NSYM$ (which symbols to count) and $NCONN$ (counting uni- and bi-directional links the same way) are not valid.

Internal validity.

The internal validity is the degree to which conclusions can be drawn about cause-effect of independent variables on the dependent variables. We dealt with the following issues:

- **Differences among subjects.** The subjects experience has approximately the same degree with regard to programming and software architecture design/modelling since most of them have at least medium experience in those areas.
- **Accuracy of subject responses.** The general and architectural understandability as well as the precision of the diagrams are rated by each subject. The responses are then subjective which represents a threat to their accuracy. But the subjects have at least medium experience in architectural design and modelling, so we consider their responses valid. We plan to consider other understandability concepts (e.g., ability of users to interpret the model correctly) in the future studies.

  From the subjects' responses it is also possible to extract some information showing that subjects did not simply tied the concepts of precision and general understandability even though we explained it in details before the study executions. Actually, there is a significant difference in precision between low and middle values of the size variables (obtained using the Kruskal-Wallis test) while the difference in the general understandability is negligible. It is also proved by significant negative linear correlation between the general understandability and the number of elements in the diagrams and its positive correlation with the expertise for the diagrams' application domain, while it is not the case with the precision. Architectural understandability deteriorates when either precision or general understandability deteriorate.
- **Other important factors.** Influence among subjects could not really be controlled. However, the study was carried out under supervision of a human observer. As know interactions between the participants have been observed, we assume that this potential threat did not affect the validity of the study.

External validity.

The external validity is the degree to which the results of the study can be generalized to the broader population under study. The following facts are identified:

- **Architectural diagrams used.** For the purpose of the study we used the architectural diagrams of different open source and industrial systems. However, even though we tried to cover different types of diagrams, our selection might be too limited or biased into a certain direction (e.g., not enough industrial systems, too much focus on UML, etc.). Again, we will address this threat to validity in our future work by replicating the study with other sets of diagrams.
- **Subjects.** In order to solve the difficulties about obtaining well-qualified subjects we used the attendees of the SHARK 2012 workshop and master students at Faculty of Computer Science of the University of Vienna. While it is possible to show that both groups have substantial experience (and also industrial backgrounds), we are aware that more empirical studies with professionals and seasoned software architects must be carried out.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented the results obtained from a study we carried out to examine to which extent the software architecture could be conveyed through architectural component diagrams, especially in relation to the precision, general understandability and architectural understandability of the diagrams. From the analysis of the data we can draw the following conclusions:

- From the correlation tests we can conclude that there is a strong linear dependency between the precision and the general understandability of the diagrams on the one hand and the architectural understandability on the other hand. That is, we can conclude that any measures that increase the general understandability and precision of architectural models directly help to improve the architectural understandability.

- The expertise for the diagrams' application domain shows a strong positive correlation with the general understandability variable. It indicates that measures to increase the domain knowledge are really helpful to increase the understanding of component models in general.

- There is a significant negative correlation between the number of elements and general understandability. This indicates that from a certain size on (in terms of number of elements), component models get hard to understand in general because of the high cognitive load and human perception limits.

- Our study also indicates that middle values of the number of components, links, elements, and symbols in the diagram significantly increase the architectural understandability compared to high or low values. The diagrams with very high numbers of elements usually suffer from mixing of several concerns which might lead to ambiguity and less precision. Very low numbers of components, links, and elements are not sufficient to model all relevant concerns of the architecture. These dependencies might also deserve to be investigated further, especially it would be interesting to indicate the thresholds of maximum (minimum) numbers of components, links, elements, and symbols that should be depicted in one diagram more precisely. So far, we consider the thresholds we found as rough indicators.

Even though this study was conducted using subjective ratings and has some other flaws indicated in the previous section, applied statistical tests and considerations are appropriate and show its feasibility. In our future work we will consider other understandability concepts (e.g., ability of users to interpret the model correctly) as well as other more complex metrics through new experiments that will help us in establishing empirically grounded guidelines supporting the design of architectural component diagrams and their evolution and maintenance in correspondence with the implemented software system.

7. ACKNOWLEDGEMENTS

This work is supported by the Austrian Science Fund (FWF), under project P24345-N23.

8. REFERENCES

[1] S. Anderson and C. Gurr. Understanding software architecture: A semantic and cognitive approach. *WICSA1*, 1999.

[2] M. A. Babar and P. Lago. Editorial: Design decisions and design rationale in software architecture. *J. Syst. Softw.*, 82(8):1195–1197, Aug. 2009.

[3] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little. *Documenting Software Architectures: Views and Beyond*. Pearson Education, 2002.

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

[5] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Trans. Softw. Eng.*, 26(8):797–814, Aug. 2000.

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

[7] C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley Professional, 2000.

[8] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, WICSA '05, pages 109–120, Washington, DC, USA, 2005. IEEE Computer Society.

[9] 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.

[10] P. Kruchten. The 4+1 view model of architecture. *IEEE Softw.*, 12(6):42–50, Nov. 1995.

[11] Y. K. Malaiya and J. Denton. Module size distribution and defect density. In *Proceedings of the 11th International Symposium on Software Reliability Engineering*, ISSRE '00, pages 62–, Washington, DC, USA, 2000. IEEE Computer Society.

[12] P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 282–292, Washington, DC, USA, 2004. IEEE Computer Society.

[13] 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.

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

[15] W. Trochim and J. P. Donnelly. *The Research Methods Knowledge Base*. Atomic Dog, Dec. 2006.

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

[17] O. Zimmermann, T. Gschwind, J. Küster, F. Leymann, and N. Schuster. Reusable architectural decision models for enterprise application development. In *Proceedings of the Quality of software architectures 3rd international conference on Software architectures, components, and applications*, QoSA'07, pages 15–32, Berlin, Heidelberg, 2007. Springer-Verlag.