Do Architectural Design Decisions Improve the Understanding of Software Architecture? Two Controlled Experiments

Mojtaba Shahin[1,2] Peng Liang*[1,3] Zengyang Li[4]

[1] State Key Lab of Software Engineering, School of Computer, Wuhan University, China
[2] Department of Computer Engineering, Neyriz Branch, Islamic Azad University, Iran
[3] Department of Computer Science, VU University Amsterdam, The Netherlands
[4] Department of Computing Science, University of Groningen, The Netherlands
mojtabashahin@gmail.com, liangp@whu.edu.cn, zengyang.li@rug.nl

ABSTRACT

Architectural design decision (ADD) and its design rationale, as a paradigm shift on documenting and enriching architecture design description, is supposed to facilitate the understanding of architecture and the reasoning behind the design rationale, which consequently improves the architecting process and gets better architecture design results. But the lack of empirical evaluation that supports this statement is one of the major reasons that prevent industrial practitioners from using ADDs in their daily architecting activities. In this paper, we conducted two controlled experiments, as a family of experiments, to investigate how presence of ADDs can improve the understanding of architecture. The main results of our experiments are: (i) using ADDs and their rationale in architecture documentation does not affect the *time* needed for completing architecture design tasks; (ii) one experiment and the family of experiments achieved a significantly better understanding of architecture design when using ADDs; and (iii) with regard to the *correctness* of architecture understanding, more experienced participants benefited more from ADDs in comparison with less experienced ones.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architecture - *Languages*, I.6.8 [**Computing Methodologies**]: Simulation and Modeling - *Visual*

General Terms

Design, Human Factors.

Keywords

Software architecture, design rationale, architectural design decision, controlled experiment, meta-analysis.

* Corresponding author

ICPC'14, June 2–3, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2879-1/14/06…$15.00.
http://dx.doi.org/10.1145/2597008.2597139

1. INTRODUCTION

With increasing size and complexity of software-intensive systems, the role of Software Architecture (SA) as a means of understanding and managing large systems has been becoming increasingly important. The high level design description of a large and complex system can help software development teams to understand and manage complicated interactions among all stakeholders of a software-intensive system [24]. SA community has devoted considerable effort to developing various approaches, techniques, and tools for improving software architecture communication among and understanding of all the key stakeholders of large-scale software-intensive systems [6].

Software architecting is a knowledge-intensive activity, in which a large amount of knowledge is being continuously produced and consumed. In the field of software architecture, a paradigm shift has occurred from describing the outcome of architecting process to describing the Architectural Knowledge (AK) created and (re)used during architecting process, including architecture design as well as the design decisions, rationale, assumptions, context, and other factors that together determine architecture solutions [11][25]. Traditional approaches to documenting SA are limited to capture this knowledge and partially result in AK vaporization [2]. The architecturally significant information is lost during architecting process, especially in architecture evaluation and maintenance activities. This situation leads to many severe problems, such as expensive system evolution, lack of stakeholders' communication, and limited reusability of architecture [2]. Architectural design decisions (ADDs) and their rationale are an important part of the AK of a system [26]. ADD is regarded as a fundamental choice which architects should make during architecture design, for example, selection of an architectural pattern [5]. Representation of ADDs as first-class entities in architecture design can alleviate aforementioned problems [2], which is supported by various ADD models and tools [5].

Although many models, tools, and guidelines have been reported about the importance of ADDs and their rationale on architecture design in the last decade [5][29], there has been limited empirical research (e.g., evaluation and measurement) on the effects of ADDs and their rationale on the understanding of SA design, which is one of the major reasons that prevents industrial practitioners from using ADDs in their daily architecting activities [43]. This fact motivated us to conduct this empirical study reported in this paper. We focused on the understanding of architecture design since understandability is one of the main factors which influence the maintainability of software design [27] and it is also considered as one of major quality characteristics that an engineering model should have [33]. An architecture design should be first understood well before any changes can be applied to it [7]. The goal of this study is to investigate whether using ADDs can improve the understanding of architecture design and the reasoning behind the design rationale. We conducted two controlled experiments with professionals and students respectively to investigate this hypothesis quantitatively in two aspects related to architecture understanding: the *time* spent to complete a given architecture design task and the *correctness* of the architecture design results, as two proxy measures [34] of architecture understanding. The original (first) experiment was conducted with experienced software designers and the second experiment (the replication of the first experiment) was conducted with SA course students. The goal of conducting two experiments, as a family of experiments, was to provide evidence for the generalization of the experiment results by repeating the experiment in different environments using different subjects but the same experiment materials. We also performed a meta-analysis to aggregate the results obtained through different individual empirical studies (i.e., experiments). By this way, we could extract more general and reliable conclusions.

The rest of this paper is organized as follows: Section 2 briefly summarizes related work on architecture understanding and how the use of ADD and design rationale has an impact on architecture design. Section 3 describes the family of experiments. Section 4 motivates the research questions, derives research hypotheses, and discusses detailed execution of individual experiments. Section 5 reports the experiment results and the accepted hypotheses. Section 6 analyzes and discusses the experiment results. Section 7 discusses the threats to the validity of this study. We conclude this paper with future work directions in Section 8.

Replication of the experiment, a meta-analysis based on the family of experiments (i.e., the original experiment and the replicated experiment), and discussion of the experimental results are the major extensions to our previous work reported in [38] and initial proposal presented in [3].

2. RELATED WORK

This section summarizes relevant work on architecture understanding and how the use of ADD and design rationale has an impact on architecture design.

Telea *et al.* conducted a review and evaluation of architecture understanding visualization tools (i.e., commercial and research tools) and techniques from a stakeholder perspective [6]. They described how visualization tools can help different stakeholders to gain more insight from software architectures. To achieve this, they distinguished three stakeholder types (i.e., technical user, project manager, and consultant) and determined what functionalities these stakeholders expect from architecture understanding visualization tools. They concluded that all type of stakeholders involved in software architecture benefit from architecture visualization techniques for architecture understanding and the visualization minimizes the waste of architecting effort [6].

Knodel *et al.* conducted a controlled experiment with 29 subjects (researchers and graduate students on software engineering) to investigate the role of configurable graphical elements in architecture understanding [7]. Configurability in this work means, for example, enabling and/or disabling certain graphical elements. The experiment results show that a different configuration of graphical elements (e.g., a few or a larger number of graphical elements) have a significant impact on static architecture analysis tasks (for example, a program comprehension task). 63% gain in effectiveness in architectural analysis tasks was achieved by changing the configuration of the graphical elements of the same tool. This finding suggests that a minor adaptation in architecture visualization tools may lead to significant improvement for architecture understanding.

Tang *et al.* empirically investigated whether a rationale-based approach could help architects to gain better design quality [13]. To achieve this, they conducted a controlled experiment with 20 designers from industry and academia. The results of the experiment show that the group equipped with design reasoning ended up with better quality design than the group without using design reasoning, however, it is revealed that there is no significant difference among the two groups in terms of time spent on doing their tasks during software design. In addition, inexperienced designers benefit more from design reasoning than experienced designers. In their experiment, both groups used the same architecture design document as experiment material, but the test group applied the rationale-based approach (i.e., design reasoning) to architecture design, while our work tries to investigate the usefulness of ADD artifacts to architecture understanding using different architecture documents (the only difference is the ADD part).

Haitzer *et al.* conducted a controlled experiment with 60 novice architects (i.e., students of software architecture course) to determine whether architectural component diagrams have any effect on architecture design understanding [8]. They employed seven architecture related questions to measure the understanding of novice architects on architecture design. The results show that the experiment group equipped with source code and architectural component diagrams had a better performance in answering the questions than the control group that only equipped with source code, in particular, the component diagrams are more helpful to understand bigger and complex architectural connections. In addition, the experiment results reveal that if the architectural guidance provided by component diagrams already existed in source code, the performance between the experiment and the control group is roughly the same. This study focuses on architecture understanding in structural viewpoint that expresses the architecture with components and connectors, while our work focuses on the impact of architecture understanding in decisional viewpoint to that of structural viewpoint considering the decisions made during architecting.

3. THE FAMILY OF EXPERIMENTS

This section describes the main characteristics of the family of experiments. The family of experiments could increase the validity of the results [23] and make the results to be generalized across the experiments [22]. A typical family of experiments consists of multiple similar experiments that pursue the same goal in order to extract significant conclusions that can be applied in practice [23]. If replications reuse original experimental planning, e.g., research questions, variables, and experimental materials, but are only different in subjects, with roughly the same background and experience; it would be categorized as *exact replication* [28]. If the subjects have different background and experience, the replications will be called *differentiated replication* [22]. As shown in Figure 1, our family of experiments is composed of two controlled experiments, the original one (Exp1) was conducted at Wuhan with subjects from academia (researchers) and industry (engineers) who had an average of 5.6 years experience on software design and development.

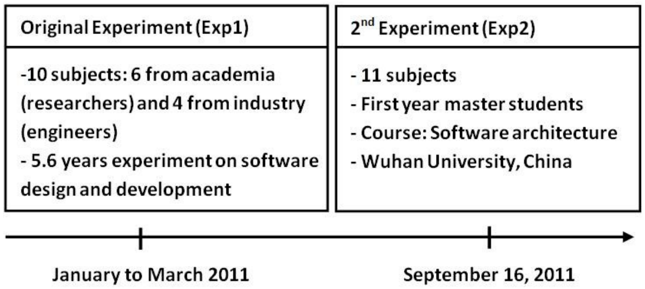
Original Experiment (Exp1)
-10 subjects: 6 from academia (researchers) and 4 from industry (engineers)
- 5.6 years experiment on software design and development
2nd Experiment (Exp2)
- 11 subjects
- First year master students
- Course: Software architecture
- Wuhan University, China
January to March 2011
September 16, 2011

Figure 1. The family of controlled experiments

The 2nd experiment (Exp2) was conducted with first year master students who had far less experience on software design and development compared to those participants in Exp1. Consequently, the Exp2 is regarded as a differentiated replication of Exp1. The main elements of the family of experiments, which are common in both experiments, are described in Table 1. Figure 1 shows that the total number of the subjects from the two experiments was 21. The original experiment (Exp1) was conducted from January to March 2011 with 10 software professionals and the 2nd experiment (Exp2) was conducted on September 16, 2011 during the software architecture course at Wuhan University, China, with 11 master students.

Table 1. Main elements of the family of experiments

Goal	To analyze the effect of using ADDs for the purpose of improving the understanding of architecture design.
Experiment Material	The sub-system architecture design and related architecture description from the architecture document of Cyber Video System (CVC).
Null Hypotheses	Using ADDs and their rationale does not affect the *time* spent to complete an architecture design task. Using ADDs and their rationale does not affect the *correctness* of the architecture design results.
Independent Variable	(none) Use of ADDs during conducting the controlled experiments.
Dependable Variables	*Time* spent on completing an architecture design task. *Correctness* of the architecture design results.

4. INDIVIDUAL EXPERIMENTS

This section provides detailed information on how the two controlled experiments were conducted according to guidelines provided in [10]. Section 4.1 describes the planning of the original experiment (Exp1) and Section 4.2 presents the 2nd experiment (Exp2) in terms of differences with Exp1.

4.1 The Original Experiment (Exp1)

4.1.1 Goal and Research Questions

The goal of the original experiment is presented in Table 1, and we planned to compare the understandability one has of an architecture design when using ADDs as opposed to without using ADDs. To this end, we need a way to evaluate and quantify the understandability someone has of an architecture design. Since architecture design understanding is crucial for an architect to perform other architecting activities (e.g., architectural maintenance and evolution), we can indirectly measure the architecture understanding by evaluating how well s/he performs architecting activities in architecting process (i.e., as a proxy measure [34]). Based on this assumption on the relationship between architecture understanding and architecting activities, the following research questions (RQs) are formulated:

- **RQ1**: Does using ADDs and their rationale reduce the *time* that is needed to complete an architecture design task as opposed to not using it?
- **RQ2**: Does using ADDs and their rationale increase the *correctness* of the architecture design results as opposed to not using it?

Associated with these two RQs, there are two null hypotheses formulated as follows:

- **$H1_0$**: Using ADDs and their rationale does not affect the *time* needed to complete an architecture design task.
- **$H2_0$**: Using ADDs and their rationale does not affect the *correctness* of the architecture design results.

The alternative hypotheses that we use in this controlled experiment are the following:

- **H1**: Using ADDs and their rationale increases the *time* needed to complete an architecture design task.
- **H2**: Using ADDs and their rationale improves the *correctness* of the architecture design results.

4.1.2 Variables

Following the standard practice of conducting controlled experiments in software engineering [10], the independent variable in this experiment is the (none) use of ADDs during conducting the controlled experiment. The experiment also consists of two dependent variables: the *time* spent by an experimental subject (e.g., a software architect) on completing an architecture design task, and the *correctness* of the architecture design results quantified by applying the evaluation criteria (see Section 5.3) on the experiment design results submitted by the subjects. The details of the experiment design and process are presented in Section 4.1.5.

4.1.3 Experimental Subjects

10 subjects participated in Exp1: 6 of them are from academia (researchers at universities), and 4 of them are from industry (developers and architects in software companies). The researchers are from two universities in Wuhan, who are major in

software engineering and development and the engineers are from three software companies. The 10 subjects have an average of 5.6 years experience on software design and development. The 10 subjects were grouped into Group A and B evenly (each group has 5 subjects) based on their expertise and background (i.e., from academia or industry). Table 2 shows the characteristics of the experimental subjects.

4.1.4 Experiment Material

The experiment material (i.e., the sub-system architecture design and related architecture description) is selected from the architecture document of Cyber Video System (CVC), which is a system that provides digital media service over a satellite connection to consumers (e.g., movies). CVC provides customers access to video content on a rent basis directly from their home and displayable on any TVs available. The movie, previously made available by CVC, is made accessible to the user as soon as a payment has been made. The reason that we select this system as the experiment material is that this system is easily understandable by the subjects without the prerequisite of any specific domain knowledge as well as the scale and complexity of the architecture design is appropriate for the duration of the experiment.

The experiment documents for Group A[1] (architecture design document with ADDs) and Group B[2] (architecture design document without ADDs) are both available online for readers reference. The only difference between the SA documents used for Groups A and B is the ADD part presented in a diagram (an example is shown in Figure 3) [15]. The SA document for Group A contains this part, and Group B does not.

The experiment results for analysis (i.e., the architecture design results by the subjects) in various formats (including Office Visio, Word, and other boxology formats) by Group A and B are also available online for experiment Exp1[3] and Exp2[4].

4.1.5 Experiment Design and Process

As presented in Section 4.1.1, the usefulness of ADD is quantified in two aspects: the *time* used to complete an architecture design task, and the *correctness* of the architecture design results. Meanwhile, to make this experiment performable, we asked the subjects to complete the experiment tasks in a limited time based on our estimation, i.e., the architecture design results should be submitted by the subjects within one and a half hours. Considering these issues, the detailed steps of this experiment are specified below (see Figure 2):

1. Divide the experimental subjects evenly into two groups (Group A and B);
2. Select an appropriate sub-system architecture design with related architecture description from an architecture document, and the selected sub-system architecture design is understandable in about 45 minutes;
3. Provide both groups a new requirement about the sub-system, and ask the subjects to make new design to satisfy this new requirement based on their understanding of the existing architecture design;
4. Group A is presented the architecture design document with ADD information in a diagram to make the new design according to the new requirement (Note that, we didn't introduce a tutorial session on what ADD is to the subjects because we thought that the ADD concepts are straightforwardly understandable. We add labels to the ADD diagram, e.g., *Design Issue*, *Positive Factor*, see the experiment material of Group A in Section 4.2 for a detailed understanding);
5. Group B is presented the architecture design document without ADD information to make the new design according to the same new requirement;
6. Ask the two groups to submit the architecture design results within one and half hour (maximum duration), with a suggestion of using maximally 45 minutes for reading the architecture document, and maximally 45 minutes for completing the architecture design task;
7. Record the actual *time* (in minutes) spent by subjects on completing the architecture design task (i.e., making new design according the requirement);
8. Evaluate quantitatively the *correctness* of the architecture design results submitted by the two groups (see Section 5.3) and finally analyze the experiment results to accept or reject the hypotheses using *t-test* (see Section 5.1).

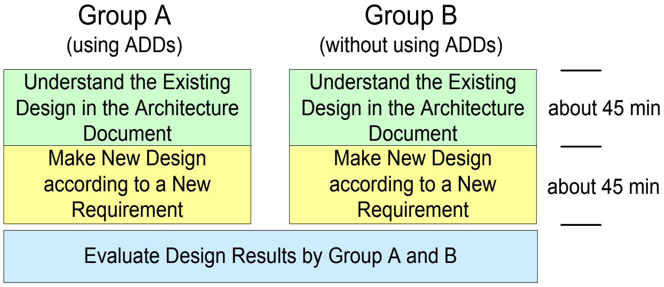
Group A
(using ADDs)
Group B
(without using ADDs)
Understand the Existing Design in the Architecture Document
Understand the Existing Design in the Architecture Document
about 45 min
Make New Design according to a New Requirement
Make New Design according to a New Requirement
about 45 min
Evaluate Design Results by Group A and B

Figure 2. Experiment design and process

Figure 2 illustrates the experiment design with a suggested maximum duration (45 minutes) for executing the individual tasks during the experiment.

4.2 The Replication Experiment (Exp2)

As mentioned in Section 3, the experiment Exp2 was a replication of Exp1 and conducted with master students in the software architecture course at Wuhan University, China, in which experimental steps, experimental material, hypotheses, and variables did not vary. As shown in Figure 1, the only difference between the two experiments Exp1 and Exp2 was the number of subjects (10 and 11 for Exp1 and Exp2 respectively) and the level of experience and background of subjects with software design and development. The subjects (first year master students) in Exp2 had far less experience than those in Exp1. This alteration

[1] This document can be downloaded at http://www.cs.vu.nl/~liangp/project/ADD/GroupA.pdf

[2] This document can be downloaded at http://www.cs.vu.nl/~liangp/project/ADD/GroupB.pdf

[3] This document can be downloaded at http://www.cs.vu.nl/~liangp/project/ADD/Exp1.zip

[4] This document can be downloaded at http://www.cs.vu.nl/~liangp/project/ADD/Exp2.zip

allowed us to analyze whether experience on software design and development affects the *correctness* of architecture design understanding and the *time* spent on completing the design task. None of the subjects participated in Exp1 and Exp2 had knowledge about ADD before they performed the experiments. Since the experiment Exp2 took place during the software architecture course, all the participants did the experiment in one place (classroom), while the participants of Exp1 were distributed in different places. It is worth nothing that Group A and Group B in Exp2 had 5 and 6 participants respectively since we have an odd number of students participated in this course. Table 3 shows the characteristics of the subjects and their results.

4.3 Meta-analysis

Like other family of experiments e.g., reported in [23][27], we considered meta-analysis as an appropriate method to aggregate the results obtained through different individual empirical studies. Meta-analysis enables us to extract more general and reliable conclusions regardless of significant results obtained or not [39]. It can investigate the global effect of a factor through the combination of different effect sizes of the experiments [27]. In our experiment; the factor is how the use of ADDs and their rationale affects the understanding of architecture design. The effect size, Cohen's "*d*", is defined as the standardized mean difference between the two groups in terms of a dependent variable [40]. Typically in software engineering, effect sizes are classified into four different values: negligible ($d < 0.2$), small ($0.2 \leq d < 0.5$), medium ($0.5 \leq d < 0.8$), and large ($d \geq 0.8$). For example, an effect size of +0.5 (medium) means that Group A scored half of a standard deviation better than Group B. In Sections 5.2 and 5.3, we describe the results of the meta-analysis based on the experiment results.

5. EXPERIMENT RESULTS

This section summarizes the results of the two controlled experiments. We first introduce the *t-test* statistical method in Section 5.1, and discuss the *time* and *correctness* results respectively in Section 5.2 and 5.3. Table 2 and Table 3 show the results (*time* spent and *correctness*) of the experiment Exp1 and Exp2 respectively. The descriptive statistics of the experiment results on *time* spent and *correctness* by Group A and B are shown in Table 4 and Table 5.

5.1 T-test

T-test is a statistical hypothesis test method that assesses if there is a significant difference between the means of two groups [12]. In statistical tests, a *p-value* represents the probability that a hypothesis test is significantly different from the null hypothesis. The *p-value* shows how likely it is that a test group is significantly different from a controlled group. To analyze the controlled experiment results, we applied the *t-test* at a significance level of 95% (α=0.05) for statistical evaluation, which means that statistical significance is attained in cases where the *p-value* is found to be lower than 0.05 or *t-test* is bigger than *t-critical*. The *t-critical value* is the cutoff between retaining and rejecting the null hypothesis. If the *t-test value* is bigger than the *t-critical value*, the null hypothesis is rejected; otherwise, the null hypothesis is retained.

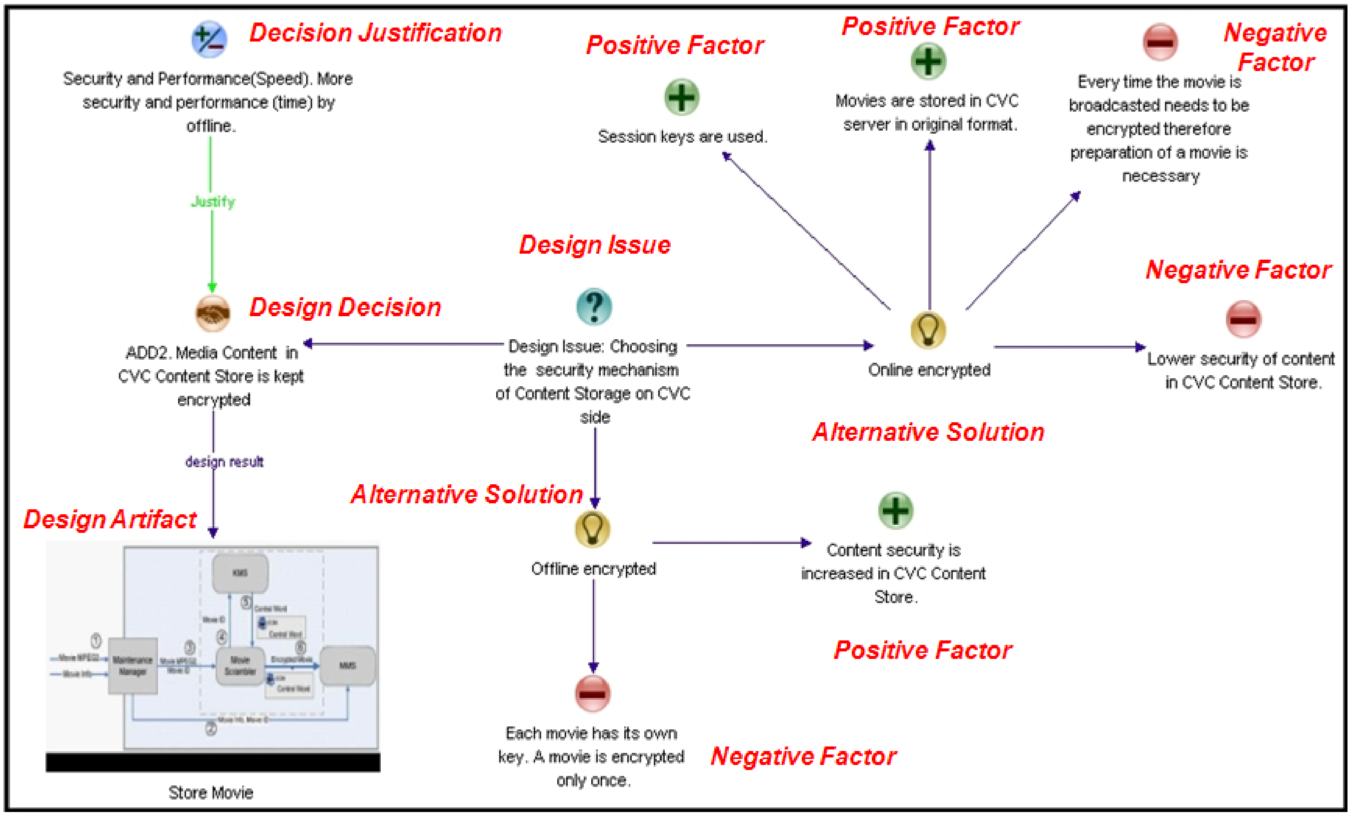
Decision Justification
Security and Performance(Speed). More security and performance (time) by offline.
Justify
Design Decision
ADD2. Media Content in CVC Content Store is kept encrypted
design result
Design Artifact
KMS
Movie MPEG2
Movie Info
Maintenance Manager
Movie MPEG2 Movie ID
Movie ID
Control Word
Movie Scrambler
Encrypted Movie
MMS
Movie Info, Movie ID
Store Movie
Design Issue
Design Issue: Choosing the security mechanism of Content Storage on CVC side
Alternative Solution
Offline encrypted
Content security is increased in CVC Content Store.
Positive Factor
Each movie has its own key. A movie is encrypted only once.
Negative Factor
Positive Factor
Session keys are used.
Positive Factor
Movies are stored in CVC server in original format.
Negative Factor
Every time the movie is broadcasted needs to be encrypted therefore preparation of a movie is necessary
Negative Factor
Online encrypted
Lower security of content in CVC Content Store.
Alternative Solution

Figure 3. Example of an ADD and its rationale presented in a diagram

Table 2. Characteristics of the subjects and their results of *time* spent and *correctness* in Exp1

Subject No.	Affiliation	Group	Total Time (min)	Correctness (point)
1	Academia	Test	44	1.75
2	Academia	Test	45	2.00
3	Industry	Test	54	1.75
4	Industry	Test	70	2.50
5	Academia	Test	65	1.75
6	Academia	Control	54	1.25
7	Industry	Control	45	1.75
8	Academia	Control	52	0.75
9	Industry	Control	90	1.75
10	Academia	Control	80	1.25

Table 3. Characteristics of the subjects and their results of *time* spent and *correctness* in Exp2

Subject No.	Affiliation	Group	Total Time (min)	Correctness (point)
1	Student	Test	41	0.25
2	Student	Test	58	1.75
3	Student	Test	55	2.00
4	Student	Test	54	2.25
5	Student	Test	56	1.00
6	Student	Control	58	1.50
7	Student	Control	55	1.50
8	Student	Control	60	1.00
9	Student	Control	52	0.25
10	Student	Control	40	1.50
11	Student	Control	52	0.25

5.2 Time Results

We started by testing the null hypothesis $\mathbf{H1_0}$, which states that using ADDs and their rationale does not affect the *time* needed to complete an architecture design task.

In the initial experiment design, it was supposed that the *time* aspect in this controlled experiment was divided into the *time* spent on reading the architecture document (time on reading activity) and *time* on making the new architecture design (time on design activity), but we noticed that the *time* duration spent for reading architecture document and design could not be fully separated (like oil and water) because the subjects could sometimes go back to read the document again during the design period, for example, to confirm/recall something which is not clear, or to verify new design with existing design. In such situation, the *time* results of "reading" and "design" provided by the subjects may not be accurate and meaningful, because it is difficult to count how much time they spent on re-reading the document during the design activity. Consequently, we consider the total *time* (inducing reading architecture document and design activity) to investigate whether there is a significant difference in the total *time* spent on completing the design task between Group A and B.

Table 4 reports a summary of descriptive statistics (i.e., mean and standard deviation (SD)) of the results of the two controlled experiments regarding *time* spent on completing an architecture design task. Firstly, we consider the data from the two experiments as one integrated experiment and the data is analyzed as one data set (the first row "Overall" in Table 4) and secondly each experiment is analyzed separately (the second and third row "Exp1"and "Exp2" in Table 4).

Figure 4 shows a comparison of the two experiments individually and together in an "Overall" perspective using box plot diagrams. These box plots show that the subjects in Group A have spent less time in completing an architecture design task in comparison to the participants in Group B. From the "Overall" row in Tables 4, we can see that the participants in Group A (using ADDs) spent less mean *time* to complete the architecture design task in comparison of Group B (respectively, 54.2 minutes and 58.0 minutes for Group A and B). This difference on *time* spent is not considered as significant between Group A and B, as *p-value* = 0.488 > 0.05. Thus, in an overall perspective, the first null hypothesis $\mathbf{H1_0}$ cannot be rejected and it means that using ADDs and their rationale does not affect the *time* needed to complete an architecture design task.

In addition, looking at the two experiments separately, we can see from Table 4 that although for Exp 1 and Exp 2 the subjects of Group A (using ADDs) spent less *time* to complete their design tasks, but the differences are not significant (respectively, *p-value* of Exp1 = 0.423 and *p-value* of Exp2 = 0.993, which both of them are greater than 0.05). We accept the null hypothesis $\mathbf{H1_0}$ for Exp1 and Exp2 and it means that using ADDs and their rationale does not affect the *time* needed to complete an architecture design task. As discussed in related work, Tang *et al.* have conducted a similar experiment to investigate the application of a design reasoning process in two groups: test group (equipped with design reasoning) and control group (without using design reasoning) [13]. Their results show that both groups took a similar amount of time to finish their tasks of software design, which is similar to our results of *time* aspect.

Concerning the effect size, we obtained Cohen's d = -0.533 for Exp1 (medium) and d = -0.004 for Exp2 (negligible). Applying meta-analysis (see Section 4.3), the mean effect size known as $d+$ [44] is -0.252, which can be considered as a small effective size. The conclusion is that using ADDs and their rationale in architecture documents does not increase the *time* spent for completing an architecture design task.

Table 4. Descriptive statistics of the experiment results on *time* spent by Group A and Group B

	Group A		Group B		p-value	t-test
	Mean	SD	Mean	SD		
Overall	54.20	9.11	58.00	14.62	0.488	0.705

Exp1	55.60	11.67	64.20	19.60	0.423	0.840
Exp2	52.80	6.76	52.83	7.05	0.993	0.007

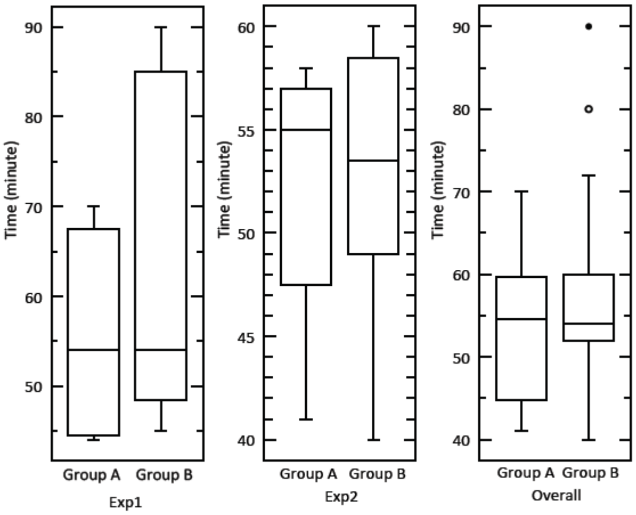

Figure 4. Box plot for *time* spent

5.3 Correctness Results

This subsection presents the test of null hypothesis $\mathbf{H2_0}$, which states that using ADDs and their rationale does not affect the *correctness* of the architecture design results. As discussed in Section 4.1.1, evaluation of *correctness* of the architecture design results acts as a proxy measure to architecture understanding. We considered three criteria, which were carefully designed to reflect one's understanding of the existing architecture design and were used to evaluate the architecture design results. The design results were scored by the first two authors separately and any disagreements among them were resolved through discussions. The score for fully satisfying each criterion is 1.0 point.

1) **Can the design result satisfy the new requirement?** This criterion is divided into four sub-criteria that address various aspects of the new requirement. Each of them has 0.25 point. For example, one sub-criterion is that "*Checking whether the request user ID is a user with Certification Authority*".

   Rationale: The reason we include this criterion is that a new architecture design that can satisfy the new requirements is based on the *correct* understanding of existing architecture design [6].

2) **Has the design result any conflict with existing design?** A design conflict refers to the situation when the new architecture design and existing design are incompatible with each other. For example, in the existing design for requirement "*Store Movie*", the movie content is encrypted before being stored in Movie Management System (MMS). If a new design for requirement "*Handle Movie Request*" doesn't consider the decryption of movie content before sending it to the movie requester, then there is a design conflict between the new design (for requirement "*Handle Movie Request*") and existing design (for requirement "*Store Movie*"). If we identify a design conflict between new design and existing design, we subtract 0.25 point for each design conflict from the full score 1.0 of this criterion.

   Rationale: We introduce this evaluation criterion since design conflict is a sign to demonstrate that the subject didn't understand the existing design *correctly* [35].

3) **Does the design result reuse any part of existing design?** If the subject uses a new component, we add 0.00 point to the score of the design result. If the subject reuses an existing component appropriately, we add 0.25 point to the score of the design result.

   Rationale: We introduce this criterion because effective and appropriate reuse of existing design artifacts is an important sign to show that the subject has a *correct* understanding of the existing design [36].

The *correctness* hypothesis $\mathbf{H2_0}$ is investigated using *t-test* based on above evaluation criteria and the evaluation results. A comparison of the two experiments, as well as in an "Overall" perspective, is shown in Figure 5 using box plots. These box plots show that the subjects in Group A have outperformed in *correctness* of understanding in comparison to the participants in Group B. Table 5 shows that the mean score of the architecture design results of Group A are higher in an overall perspective, as well as in separated experiment (Exp1 and Exp2). In comparison to Figure 4, Figure 5 shows that the difference between Group A and B in terms of *correctness* is more obvious than for the *time* aspect. Table 5 shows that the mean score of Group A is higher than Group B for "Overall" (47%) as well as for Exp1 (44%) and Exp2 (45%). The *t-test* result provides evidence that the difference between the two groups, in terms of *correctness* of architecture design results (a proxy measure of architecture understanding), is significant in an "Overall" perspective (*p-value* = 0.049 < 0.05) and for experiment Exp1(*p-value* = 0.035 < 0.05). Therefore, we can reject the null hypothesis $\mathbf{H2_0}$ and accept the alternative one (i.e., **H2**) for the "Overall" experiment and Exp1. Whereas for Exp2, which involves less experienced subjects on software design and development, although the difference is in favor of the group that uses ADDs (Group A), but this difference is not considered as significant (*p-value* = 0.320 > 0.05). Thus, the null hypothesis $\mathbf{H2_0}$ cannot be rejected for Exp2. This *correctness* result implies that more experienced participants benefit more than less experienced ones when they use ADDs, as for their *correctness* of architecture design results.

Concerning the effect size, we obtained Cohen's d = 1.630 for Exp1 (large) and d = 0.627 for Exp2 (medium). Applying meta-analysis (see Section 4.3), we can obtain the mean effect size $d+$ is 1.04, which can be considered as a large effect size. In general, it can be concluded that using ADDs and their rationale can substantially improve the *correctness* of architecture design understanding.

5.4 Summary of the Results

To summarize the main results, it is observed that presence of ADDs and their rationale does improve the *correctness* of architecture design results, but does not affect the *time* needed for completing an architecture design task in all experiments (i.e., Exp1, Exp2, and the "Overall" experiments).

In terms of statistically significant differences between the results of control and test groups, the difference on *correctness* is significant in Exp1 and the “Overall” experiment, but insignificant in Exp2; the difference on *time* spent is not significant in all experiments (i.e., Exp1, Exp2, and the “Overall” experiments).

Table 5. Descriptive statistics of the experimental results on *correctness* by Group A and Group B

	Group A		Group B		p-value	t-test
	Mean	SD	Mean	SD		
Overall	1.70	0.64	1.15	0.53	**0.049**	2.095
Exp1	1.95	0.32	1.35	0.41	**0.035**	2.530
Exp2	1.45	0.81	1.00	0.61	0.320	1.041

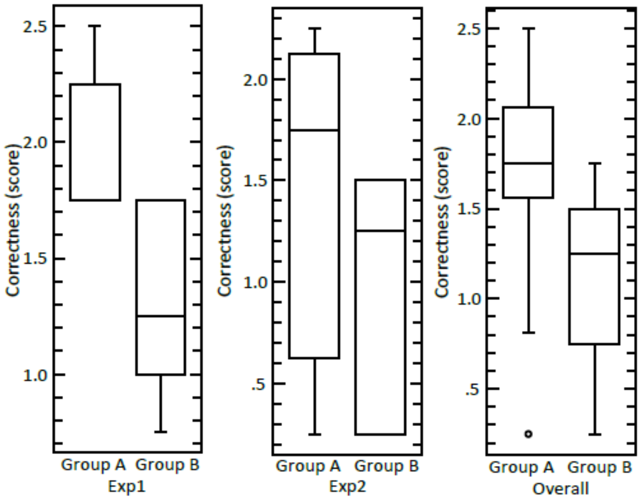

Figure 5. Box plot for *correctness* of design results

6. DISCUSSION

The main goal of this study was to analyze how architectural design decisions (ADDs) and their rationale influence the understanding of software architecture. We analyze and discuss the results obtained from the experiments in three aspects: explanation of the results, the impact of participants’ experience to the results, and the challenges of evaluating architecture understanding.

6.1 Explaining the Obtained Results

The experiment result shows that participants equipped with ADDs and their rationale achieved a better *correctness* of architecture understanding in comparison of those without presence of ADDs. This is true in two experiments and the “Overall” experiment, and the difference is statistically significant in Exp1 (professionals experiment) and the “Overall” experiment except for Exp2 (students experiment), which implies that there is a difference on the effect of ADD to the *correctness* of architecting understanding between participants who have different levels of software design experience (more dedicated discussion in Section 6.2). Regarding the evaluation criteria introduced in Section 5.3 to measure the *correctness* of the architecture design results by the subjects, it is worth noting that test group (i.e., using ADDs) outperformed consistently in all criteria than the control group (without using ADDs). For *time* spent on completing assigned architecture design task, the results confirmed that employing ADDs and their rationale does not increase the *time* needed to accomplish the design task. Consequently, the difference on *time* aspect is not statistically significant in both experiments as well as in the “Overall” experiment.

6.2 The Impact of Participants’ Experience

As described in Section 4, the participants (professionals) in Exp1 had more experience than those in Exp2 (students). The result of *correctness* reveals that more experienced participants benefit more (i.e., significant difference in Exp1) than less experienced participants (i.e., insignificant difference in Exp2) when they use ADDs, as for their *correctness* of architecture design results. This result (the effect of ADD to architecture design understanding) is contradictory to the result reported in [13] (the effect of design reasoning process to architecture design quality), which shows that inexperienced designers benefit more from design reasoning than experienced designers. The results of our work and [13] seem to be different at the first sight, but actually are complementary to each other, which imply that inexperienced (e.g., novice) designers need to be trained and equipped with design reasoning process to make the most use of ADDs [42].

The experiment result of *correctness* also shows that there is no significant difference (*p-value* = 0.814 > 0.05) between less experienced participants using ADDs (Group A students in Exp2) and more experienced participants without using ADDs (Group B professionals in Exp1), which indicates that presence of ADDs can do assist less experienced designers achieve an equally *correct* understanding of architecture comparable to experienced designers (without using ADDs), which is a kind of way to remedy their deficiencies in design experience.

For the *time* aspect, one unexpected result is that less experienced participants (students in Exp2) spent less mean *time* (e.g., 52.83 versus 64.20 minutes for Group B) on architecture design task than experienced participants (professionals in Exp1), which partially due to the reason that Exp2 (students experiment) was conducted for all the participants during a course, which might pose time pressure to some participants when others finished the experiment task and submitted the results earlier, which is not the case for Exp1 that was conducted one by one.

6.3 Evaluating Architecture Understanding

Three challenges have been introduced to empirically evaluate understandability or comprehensibility of designs (e.g., models): information equivalence, accessibility of participants, and researcher bias [41]. The first challenge, information equivalence, was not happened in our experiment because the two groups (control and test groups) in both experiments use the same experiment material, and the only difference between the architecture documents used for the two groups is the ADD part that is presented in a diagram. The challenge of accessibility of participants (i.e., finding competent participants on architecture) was not happened in Exp1 because the participants in Exp1 had sufficient expertise and experience in software design and development. In Exp2, we used master students of software architecture course as subjects, who represent novice architects and have basic skills of designing architecture given in the course. Lastly, we could not eliminate the third challenge, researcher bias (i.e., the evaluator of a notation is often its proponent), because the researchers (authors) of this work were involved in the evaluation of experiment design results.

7. THREATS TO VALIDITY

Due to the limitations of this controlled experiment, there are several threats to the validity of the experiment results. We classify them into threats to construct validity, which refers to the degree to which the measures in a study actually represents the constructs in the real world; internal validity, which describes the cause-effect relationship between the independent and dependent variables; external validity, which refers to the ability to generalize the results and findings of the experiments in different contexts; and conclusion validity, which refers to the degree to which the conclusion that is being achieved from the experiment is reasonable [18].

7.1 Construct Validity

In our study, the construct validity is related to how the *time* spent on completing an architecture design task and the *correctness* of architecture understanding were accurately measured by their appropriated instruments. In terms of *time* spent on completing the architecture design task, it was measured by time sheet and was monitored and checked by the second author, who was present during the two experiments. Time sheet is a common practice for measuring the time required to do experiment tasks in controlled experiments [23]. Understanding of architecture was measured by evaluating the *correctness* of architecture design results. Unlike the methods used in [22][23], we did not use questionnaires to measure the understanding of the subjects on architecture design. As discussed in Section 5.3, we used an evaluation model (with three criteria), which was constructed in advance and was peer-reviewed by the authors, to evaluate the *correctness* of the architecture design results. These evaluation criteria were carefully designed to reflect one's understanding of the existing architecture design, in order to mitigate the threat to construct validity.

7.2 Internal Validity

One of the usual internal validity threats about the experimental subjects is that the knowledge of subjects may differ between the test group and controlled group, and possibly affect the quality of experiment results. To alleviate this threat, in Exp1, the subjects were evenly distributed across the groups based on their expertise and background (e.g., from academia and industry, see Table 2). In Exp2, the participants were randomly distributed and their background was quite similar in which about 90% of them didn't have any industry experiences. The subjects participated in Exp1 had enough competency to do the experimental design task because they have being working on software (architecture) design and development for several years, including academic researchers and industrial practitioners. The subjects of Exp2 had enough motivation to participate in this experiment because the architecture design task was part of their SA course. None of participants knew the goal and hypotheses of this experiment before performing the experiment, which partially mitigates the bias of participants when completing the architecture design task.

Researchers' bias as a validity threat can affect the results of the study. This threat exists in our study because the *correctness* of the experiment design results (see Section 5.3) were evaluated and graded by two researchers who were involved in this experiment (i.e., the first two authors of this paper). External evaluators without the knowledge of the goal and hypotheses of this experiment can be used to eliminate this threat, but we failed to find suitable evaluators largely because of the considerable effort needed to evaluate all the design results using the evaluation criteria (see Section 5.3). In order to alleviate this threat, an evaluation model to the design results (the criteria introduced in Section 5.3) was designed in advance that clearly states the required design elements and the corresponding points for each element, and these criteria were agreed by the researchers who evaluated the design results. Furthermore, to have a reliable and objective evaluation, all the answers of subjects were evaluated and graded by two researchers independently. Next, the grading results were compared and any disagreements were resolved through discussions.

7.3 External Validity

The total number of subjects (21) participated in two experiments is limited due to the criteria for selecting the experimental subjects and the resources we have: we could only use experienced software developers and architects in Exp1. The considerable effort of participants (one and a half hours) for performing this experiment is another difficulty to include more experienced subjects since this experiment is not directly related with their job. We used master students as subjects in Exp2. We regard these master students as next generation of software professionals in this experimental context [31] since Host *et al.* found that students are suitable replacements for industry professionals if performing small tasks of judgment [32]. We plan to repeat another experiment with more subjects in our future work by introducing ADD in some industrial projects.

The number of architecture evaluators (two), i.e., the first two authors of this paper for evaluating the design results is quite limited since experienced evaluators are scarce resources and the evaluation against the criteria takes considerable time. The involvement of more evaluators can reduce the bias in the evaluation of architecture design results. We tried to alleviate this threat by a pilot evaluation and intensive discussions between the two evaluators to reach consensus during evaluation.

The size of the architecture design task used in this experiment is relative small due to the time limitation of the experiment. In both experiments, the subjects had one and a half hours to complete their assigned design tasks. However, we asked the subjects to complete the design task as soon as they can. We plan to extend the task of this experiment to a real architecture project (for example, an architecture course project or architecture design in an industry project) that covers an integrated architecting process (i.e., architectural analysis, synthesis, and evaluation) and includes the collaboration and communication between architects and involved stakeholders, in order to investigate the systematic use of ADDs in architecting.

7.4 Conclusion Validity

In this experiment, we use *t-test* (see Section 5.1), a parametric test requiring that each of the two populations being compared should follow a normal distribution. This condition can be tested using, e.g., Shapiro-Wilk test [19]. The Shapiro-Wilk test succeeded for *time* spent and *correctness* of the experimental results, which means that *t-test* can be used. Another threat to the conclusion validity is the analysis and discussion of the data aggregated (i.e., “Overall”) from the two experiments. The reason for this aggregation was that each experiment included a small number of subjects, whereas aggregating data with larger sample subjects could contribute to the statistical robustness and generalizability of the conclusions [23]. However, aggregating data from different individual experiments may have a negative impact to the conclusion validity due to differences between the settings of the experiments and the groups of subjects. This threat is not so serious since the experiment results revealed that the two experiments have had roughly similar results separately, and the only difference (if any) was in the level of significance, which is to be expected in small numbers of subjects.

8. CONCLUSIONS AND FUTURE WORK

In software architecture community, ADD has been recognized as a first-class element in software architecture [2], as well as in architectural description standard [11]. ADD and its design rationale is supposed to provide an intuitive way for architects and concerned stakeholders to communicate, use, and share ADDs, and consequently facilitates the understanding of architecture design, but there is no empirical evaluation that supports this statement. Two controlled experiments were conducted in this work, and the experiment results indicates that: (1) using ADDs and their rationale in architecture documentation does not affect the *time* needed for completing architecture design tasks; (2) the test group, which was provided with ADDs in an architecture document, produced significantly better architecture design results than the controlled group in one experiment and the family of experiments; (3) regarding the *correctness* of architecture understanding, the participants with more design experience gained more benefit from using ADDs in comparison with less experienced participants.

We outline our ongoing and future work in the following directions: (1) Conduct new controlled experiments on different systems with more experimental subjects and external design results evaluators in order to address the limitations discussed in Section 7 and achieve more convincible and generalizable experiment results. (2) Conduct a controlled experiment in an integrated architecting process, which covers all the architecting activities, including architectural analysis, synthesis, and evaluation. The purpose of the new experiment is to investigate qualitatively and quantitatively the usefulness and effectiveness of ADD and design rationale in an architecting lifecycle. (3) The prerequisite of using ADDs in architecting process is to capture or recover ADDs and the traceability links from and to an ADD in architecture documents [37], which requires considerable effort (including manual or semi-automatic work with the support of ADD tools). We plan to further investigate the effort of creating, recovering, and using ADDs for the cost-benefit analysis of introducing ADD in architecting.

9. ACKNOWLEDGMENTS

This work has been partially supported by the Natural Science Foundation of China (NSFC) under Grant No.61170025 and AFR-Luxembourg under the contract No.895528. We would also like to thank all the participants of the two controlled experiments, and insightful comments for this work by Muhammad Ali Babar, Paris Avgeriou, and Anton Jansen.

10. REFERENCES

[1] Kruchten, P., Lago, P., and van Vliet, H. 2006. Building up and reasoning about architectural knowledge. In *Proceedings of the 2nd International Conference on the Quality of Software-Architectures (QoSA)*, Springer LNCS, 43-58.

[2] Jansen, A. and Bosch, J. 2005. Software architecture as a set of architectural design decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, 109-120.

[3] Shahin, M., Liang, P., and Mohammad, R. K. 2010. Improving understandability of architecture design through visualization of architectural design decision, In *Proceedings of the 5th Workshop on SHAring and Reusing architectural Knowledge (SHARK)*, ACM, 88-95.

[4] Liang, P. and Avgeriou, P. 2009. Tools and Technologies for Architecture Knowledge Management. In *Software Architecture Knowledge Management: Theory and Practice*, Springer, 91-111.

[5] Shahin, M., Liang, P., and Mohammad, R. K. 2009. Architectural design decision: Existing models and tools. In *Proceedings of the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, 293-296.

[6] Telea, A., Voinea, L., and Sassenburg, H. 2010. Visual tools for software architecture understanding: A stakeholder perspective, *IEEE Software*, 27(6):46-53.

[7] Knodel, J., Muthig, D., and Naab, M. 2008. An experiment on the role of graphical elements in architecture visualization, *Empirical Software Engineering*, 13(6):693-726.

[8] Haitzer, T. and Zdun, U. 2013. Controlled experiment on the supportive effect of architectural component diagrams for design understanding of novice architects, In *Proceedings of the 7th European conference on Software Architecture (ECSA)*, Springer LNCS, 54-71.

[9] Ugai, T., Hayashi, S., and Saeki, M. 2010. Visualizing stakeholder concerns with anchored map, In *Proceedings of the 5th International Workshop on Requirements Engineering Visualization (REV)*, IEEE, 20-24.

[10] Jedlitschka, A. and Pfahl, D. 2005. Reporting guidelines for controlled experiments in software engineering. In *Proceedings of the 4th International Symposium on Empirical Software Engineering (ISESE)*, IEEE, 95-104.

[11] ISO, 2011. ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description, http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

[12] Ruxton, G. D. 2006. The unequal variance t-test is an underused alternative to Student's t-test and the Mann-Whitney U test. *Behavioral Ecology*, 17(4):688-690.

[13] Tang, A., Tran, M. H., Han, J., and van Vliet, H. 2008. Design reasoning improves software design quality, In *Proceeding of the 4th International Conference on Quality of Software-Architectures: Models and Architectures (QoSA)*, Springer LNCS, 28-42.

[14] Jansen, A., van der Ven, J., Avgeriou, P., and Hammer, D. K. 2007. Tool support for architectural decisions, In *Proceedings of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, 4-13.

[15] Shahin, M., Liang, P., and Mohammad, R. K. 2010. Rationale visualization of software architectural design decision using Compendium, In *Proceedings of the 25th ACM Symposium on Applied Computing (SAC)*, ACM, 2358-2359.

[16] Mirakhorli, M. 2011. Tracing architecturally significant requirements: A decision-centric approach, In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, ACM, 1126-1127.

[17] Kunz, W. and Rittel, H. 1970. *Issues as Elements of Information Systems*. Studiengruppe fur Systemforschung.

[18] Wohlin, C., Host, M., and Henningsson, K. 2003. Empirical research methods in software engineering. *Empirical Methods and Studies in Software Engineering*, Springer, 145-165.

[19] Royston, P. 1982. An Extension of Shapiro and Wilk's W Test for Normality to Large Samples, *Applied Statistics*, 31:115-124.

[20] Tang, A. and Lago, P. 2010. Notes on Design Reasoning Techniques (V1.4), SUTICT-TR2010.01, Swinburne University of Technology & VU University Amsterdam.

[21] Basili, V. R., Shull, F., and Lanubile, F. 1999. Building knowledge through families of experiments, *IEEE Transactions on Software Engineering*, 25(4):456-473.

[22] Reggio, G., Ricca, F., Scanniello, G., Di Cerbo, F., and Dodero, G. 2013. On the comprehension of workflows modeled with a precise style: Results from a family of controlled experiments, *Journal on Software & Systems Modeling*, DOI:10.1007/s10270-013-0386-9.

[23] Hadar, I., Reinhartz-Berger, I., Kuflik, T., Perini, A., Ricca, F., and Susi, A. 2013. Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments, *Information and Software Technology*, 55(10):1823-1843.

[24] Bass, L., Clements, P., and Kazman, R. 2012. *Software Architecture in Practice, 3rd Edition*, Addison Wesley.

[25] Kruchten, P., Lago, P., van Vliet, H., and Wolf, T. 2005. Building up and exploiting architectural knowledge. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, 291-292.

[26] Babar, M. A., Dingsøyr, T., Lago, P., and van Vliet, H. 2009. *Software Architecture Knowledge Management: Theory and Practice*. Springer-Verlag New York.

[27] Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., and Carsí, J. A. 2011. Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments, *Information and Software Technology*, 53(12):1391-1403.

[28] Shull, F., Carver, J. C., Vegas, S, and Juristo, N. 2008. The role of replications in empirical software engineering, *Empirical Software Engineering*, 13(2):211-218.

[29] Li, Z., Liang, P., and Avgeriou, P. 2013. Application of knowledge-based approaches in software architecture: A systematic mapping study, *Information and Software Technology*, 55(5):777-794.

[30] Sjoberg, D. I. K., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N., and Rekdal, A. C. 2005. A survey of controlled experiments in software engineering, *IEEE Transactions on Software Engineering*, 31(9):733-753.

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

[32] Host, M., Regnell, B., and Wohlin, C. 2000. Using students as subjects - A comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201-214.

[33] Selic, B. 2003. The pragmatics of model-driven development, *IEEE Software*, 20(5):19-25.

[34] Rello, L. and Baeza-Yates, R. 2012. Lexical quality as a proxy for web text understandability, In *Proceedings of the 21st International Conference Companion on World Wide Web (WWW)*, ACM, 591-592.

[35] Lu, S. Y., Cai, J., Burkett, W., and Udwadia, F. 2000. A methodology for collaborative design process and conflict analysis, *CIRP Annals-Manufacturing Technology*, 49(1):69-73.

[36] Girczyc, E. and Carlson, S. 1993. Increasing design quality and engineering productivity through design reuse, In *Proceedings of the 30th international Design Automation Conference (DAC)*, ACM, 48-53.

[37] Mirakhorli, M. and Cleland-Huang, J. 2011. Transforming trace information in architectural documents into re-usable and effective traceability links, In *Proceedings of the 6th Workshop on SHAring and Reusing architectural Knowledge (SHARK)*, ACM, 45-52.

[38] Shahin, M., Liang, P., and Li, Z. 2011. Architectural design decision visualization for architecture design: Preliminary results of a controlled experiment, In *Proceedings of the 5th European Conference on Software Architecture: Companion Volume (ECSA)*, ACM, 2-9.

[39] Pickarda, L. M., Kitchenhama, B. A., and Jonesb, P. W. 1998. Combining empirical results in software engineering, *Information and Software Technology*, 40(14):811-821.

[40] Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences, 2nd Edition*, Lawrence Earlbaum Associates, Hillsdale, NJ.

[41] Aranda, J., Ernst, N., Horkoff, J., and Easterbrook, S. 2007. A framework for empirical evaluation of model comprehensibility, In *Proceeding of Workshop on Modeling in Software Engineering (MiSE)*, IEEE, 7-13.

[42] van Heesch, U. and Avgeriou, P. 2010. Naive architecting - Understanding the reasoning process of students - A descriptive survey, In *Proceedings of the 4th European conference on Software Architecture (ECSA)*, Springer LNCS, 24-37.

[43] Lago, P., Avgeriou, P., Capilla, R., and Kruchten, P. 2008. Wishes and boundaries for a software architecture knowledge community. In *Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, 271-274.

[44] Hedges, L. V, and Olkin, I. 1985. *Statistical Methods for Meta-analysis*. Academic Press Orlando.