Assessment of a Framework for Designing and Evaluating Security Sensitive Architecture

Muhammad Ali Babar

Lero, University of Limerick, Ireland
malibaba@lero.ie

Background: We have developed an Architectural Level Security Analysis Framework (ALSAF), which can be used to consider and address security related issues at software architecture level.

Goal: Our goal was to empirically assess the usefulness of ALSAF for identifying security attributes and security design patterns for satisfying those attributes during architecture design and evaluation.

Assessment approach: The reported assessment was performed with one pilot study and one Quasi-experiment. In the main study, there were 19 software development professionals who participated in the study after attending a training course. The participants were required to identify security attributes and security design patterns suitable for achieving those attributes based on a given list of security properties. One group (control group) was given the textual description of security patterns, attributes, and properties, the other group (treatment group) was given ALSAF as well as the document provided to the control group. The outcome variables were security attributes and security patterns for a Web-based system, whose requirements were provided to the participants.

Result: The average score for identifying security attributes for the treatment group was 4.56 and for the control group was 2.60. The difference between the groups was significant using Mann-Whiney test (p=0.011). The average score for identifying the security patterns for the treatment group was 5.78 and for the control group was 2.8. Mann-Whitney test revealed that the difference between the groups was again significant at (p=0.022). Post-study questionnaire revealed that majority of the participants were convinced of the usefulness of ALSAF in identifying and understanding the relationships between security attributes, properties, and patterns for supporting architectural level security analysis.

Conclusion: The findings provide an initial evidence to support the claim of the usefulness of ALSAF for supporting security sensitive analysis during architecture design and evaluation.

Keywords: Software architecture evaluation, security attributes, technology assessment, empirical studies

1. INTRODUCTION

Quality is one of the most important issues in software development. It has been shown that software architecture (SA) greatly influences the achievement of various quality attributes (such as security, performance, maintainability and usability) in a software intensive system [1]. Security has become one of the most important quality attributes required in networked applications, which support business- and mission-critical processes. The reported cases of security breaches have been growing exponentially over the past decade [2]. Despite the increasing importance of the security, many systems are usually architected and implemented without paying sufficient attention to security related issues, which are often not dealt with until late in the development lifecycle [3-5]. We assert that one of the major reasons for insufficient attention to security issues during software architecture design and evaluation is that many software designers do not have the required security expertise [6]. Moreover, software designers and security engineers have different preferences [7]. Security experts are primarily concerned with security, while software designers need to consider not only security but other quality attributes (such as performance, usability and maintainability) as well. Furthermore, knowledge about techniques addressing security issues has not been captured and documented in a format that is easily accessible and understandable for designers, which makes engineering for security early in the design process difficult [5, 8].

We have developed an Architectural Level Security Analysis Framework (ALSAF) for systemically considering and addressing those security issues that need architectural level support [6]. This framework presents a set of security attributes and properties along with the design patterns known to satisfy them in an integrated format. We assert that ALSAF can help identify those security sensitive solutions, which cannot be cost-effectively retro-fitted into an implemented system - rather such solutions can only be easily applicable during the architecture

design and evaluation stages. The ALSAF can be an important source of architecturally sensitive security knowledge to help understand the relationship between software architecture and security related quality attributes and to improve the architecture design and evaluation processes for security sensitive systems.

In order to empirically assess the usefulness of ALSAF, we decided to carry out an empirical research project consisting of a set of experiments. Our research project was developed based on the guidelines provided in [9, 10]. The research project also included a pilot study aimed at assessing the experimental design, material, and logistics. This paper reports the logistics and results from the pilot and one of the main studies. In the next section, we discuss the issues that motivated this research and present ALSAF. Section 3 provides a detailed description of the design and logistics involved in the empirical assessment of ALSAF. Section 4 presents the results. Section 5 summarises and concludes this paper.

2. BACKGROUND AND MOTIVATION

A quality attribute is a non-functional requirement of a software system, such as reliability, modifiability and performance. Like many other researchers [11, 12], we believe that quality attributes of large software intensive systems are usually determined by the system's software architecture. Since software architecture plays a vital role in achieving quality attributes, it is important to address quality attributes during the software architecture process. It has also been reported that software architectures of complex and large systems are usually developed using many different patterns [13]. One of the main goals of using patterns is to design an architecture with known quality attributes [14] as each pattern either supports or inhibits one or more quality attributes. Security patterns aim to document the strengths and weaknesses of different approaches to design security sensitive systems [5, 7].

We believe that architectural decisions made by taking security into consideration can sufficiently address most of the security related problems commonly observed in many systems. For example, it is possible to modify or integrate the security policy late in the development and avoid huge code rewriting by using a suitable security sensitive architecture pattern like check point [15]. However, security expertise embodied by security patterns and its significance for software architecture is not well understood outside the security community. Moreover, software designers usually do not have access to an integrated set of solutions to security issues that needs to be addressed at the software architecture level [5].

Apart from our own work on discovering architecture knowledge from patterns [16-18], the idea of developing a framework that can help understand the relationship between software quality attributes and software architecture has been inspired by the work of [19, 20] on linking usability and software architecture. To support the design and evaluation of security sensitive architectures, we decided to systematically analyze existing security patterns in order to identify those patterns which have architectural implications for security attributes. Since we found no standard definition of the security quality attribute, we identified the most common interpretations of security and grouped them in a small set of security attributes [6]. We found that drawing a direct relationship between quality attributes and patterns was extremely hard. That was why we decomposed the identified security attributes into more detailed elements, properties, which can be considered a form of high level requirements as we can use general scenarios to characterize them [21]. Then we established the high level relationships among the identified security quality attributes, their properties, and security patterns. To identify the patterns, quality attributes, and properties, establishing relationships among them, and assessing the correctness of each step followed, we used several approaches, which have been described in [6].

2.1 The Architecture Level Security Analysis Framework

In this section, we briefly describe the architectural level security analysis framework, which has been fully described in [6]. The security framework, ALSAF, is shown in Figure 1. ALSAF consists of three layers: attributes, properties, and patterns. ALSAF is aimed at supporting systematic reasoning about security issues at the architecture level by explicating the relationships among security sensitive attributes, properties, and patterns. ALSAF presents a collection of security related attributes, properties, and patterns along with the links that form the relationships between software architecture and security quality attribute. ALSAF is expected to be used as a high level guidance to aid in analysing the architecturally sensitive security issues and their potential solutions. A security attribute is characterized by one or more security properties, which specify security requirements using scenarios, and patterns are used to satisfy those scenarios and in turn properties. Thus, security patterns provide a mechanism to bridge the gap between the problem and solution domains [6]. ALSAF also demonstrates that the relationships between patterns, properties and attributes are not necessarily binary. Nor are the relationships only positive, however, to keep the diagram uncluttered, only positive relationships have been shown. For example, the "Limited View" pattern has a negative relationship with the "Guidance" property. This is because different types of guidance material need to be provided to different categories of users of a system, which can increase the cognitive load for a user who belongs to many classes of users. Moreover, it is difficult to implement it [8].

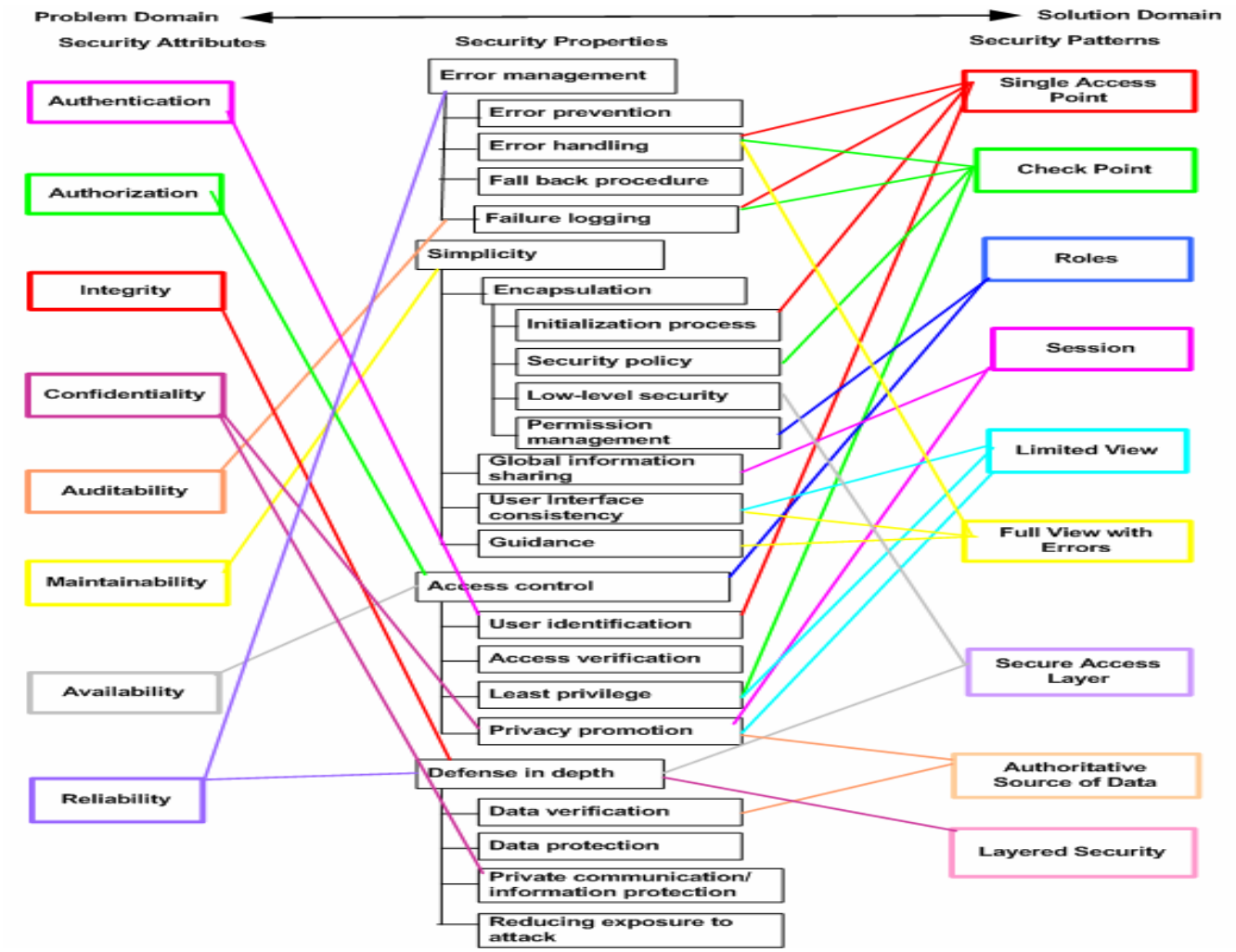

Figure 1: The Framework for supporting security sensitive architecture design and evaluation

2.2 Expected Usage of the Framework

We assert that this framework can be used by a designer or a team of designer in several ways. For example, a designer may find a certain security property vital for the secure operations of a system. He/She may realize that the importance of a particular property either by looking at the framework, from the stakeholders' scenarios, or from studying a similar system's properties. Having realized that a particular security property is important for the system to be designed, the designer can use the framework for identifying the potential patterns that need to be introduced and the security attributes that will be affected (positively or negatively). For example, to develop an online virtual training system, various levels of access are required depending upon the status of a user (such as trainers, students, coach and others). Having realized the necessity of an appropriate access control mechanism, a designer can use ALSAF, which shows that the "Roles" pattern is linked to the "access control" security property, which in turn is linked to four security attributes (i.e. authentication, authorization, integrity, and availability). That means if the designer needs to achieve the "access control" security property, he/she can introduce the "Roles" pattern to support the "access control" property. However, the "Roles" patterns needs to be supported by the "single access point", "check point", "session", "Limited view", and "authoritative source of data" patterns to achieve all four security attributes linked with the "access control" property.

In another scenario, a designer may find that the "auditability" attribute is required. The designer can use the security framework to identify the property that is needed to achieve that attribute, namely "Failure logging" in this case. Having identified the security property of "auditability", the designer can use ALSAF to find out that the "Single Access point" and "Check point" patterns are needed in order to achieve the "auditability" quality attribute. Moreover, ALSAF can also help a designer to understand why those two patterns need to be used in tandem and which other security properties and attributes are expected to be achieved by using all these patterns together.

In a third scenario describing the potential use of ALSAF, a software architecture evaluation team may find out that a particular pattern has been used in the architecture being evaluated. They can use ALSAF to identify the

properties and attributes that are supported by that pattern. Since security properties are non-functional requirements, ALSAF can help the evaluation team appreciate the security related non-functional requirements that have been considered in the architecture by using a particular security pattern. For example, if the team finds the "Authoritative Source of Data" pattern being used in the architecture being evaluated, they can use the security framework to see that this pattern promotes the "Data Verification" security property, which is positively related to the "Integrity" and "Reliability" security attributes.

3. EMPIRICAL ASSESSMENT

This section presents the design and logistics of the empirical study. This study was an experiment with one treatment group and one control group. We also ran a pilot study to gain some feedback and assess the experimental material and tasks. Because of limited space, we can only provide a summary of the pilot study.

3.1 The pilot study

The pilot study was run without a control group. Rather, all the participants had access to ALSAF while performing the required tasks. Moreover, we did not have any hypothesis to be tested at that stage. Rather, we expected to refine our research questions through the pilot study. The study was run in one of the lectures on Software Architecture of Advanced Software Engineering course offered by University of New South Wales. There were 9 volunteer participants who agreed to perform the required tasks as class exercise towards the end of the lecture. There were no marks or incentives for this exercise except the potential for learning something new. The participants also knew that the researcher did not have any influence on the offered course or their final grades as the researcher was a guest lecturer. All the participants were working in the industry in software development related jobs. Because of time constraints and logistical reasons, we did not gather demographic data. But we can say that they had a good mix of skills in designing and implementing large scale systems.

The participants were required to perform the tasks (i.e., identifying security attributes and patterns required to satisfy security properties of a given system) described in Section 3.6.2. We also used the same software requirements specifications and support material prepared for the main study. The participants were required to provide their answers in the second and fourth columns of Table 4. We also asked them to give comments about the usefulness of ALSAF and suggestions for improving the material.

The answers of the participants were marked according to the scheme described in Section 3.6.2. Based on descriptive statistics for the data obtained from participants' answers, average score for identifying correct security attributes and patterns were **4.5 and 7.2 respectively with median score of 4 and 7 marks**. Almost all of the participants were convinced that ALSAF was very useful in helping them to identify the security attributes and patterns required for QVS. Many of them also provided descriptive comments about the effectiveness of the ALSAF. However, we are unable to reproduce them because of limited space. The participants were also satisfied with the level of clarity of the requirements and format of the support material. There were suggestions for minor improvements that we taken on board while finalizing the material for the main study. The results also helped us to convert our research questions in a set of hypotheses.

3.2 Research Hypotheses

Informally, our experimental hypothesis is that ALSAF makes the pattern-based security sensitive architectural knowledge available to software engineers in a format that is more helpful in gaining a high level understanding of security patterns' objectives and rationale with regards to the security quality attributes than reading that pattern's standard documentation. Hence, this study compares the pattern-based security sensitive architectural knowledge captured in ALSAF with the standard documentation of security attributes, properties, and patterns. The context for the study described in this paper is the analysis and synthesis tasks of a general model of software architecture design described in [22]. The architectural analysis is aimed at identifying required quality attributes (i.e., security attributes in this study) and the architectural synthesis identifies candidate solutions (i.e., security patterns in this study) that can satisfy the required security quality attributes.

More formally, the null hypothesis for identifying security quality attributes is:

H01: ALSAF is no more useful in identifying the required security attributes than the description of the security attributes provided in standard documentation format.

Our alternative hypothesis is:

H1: ALSAF is more useful in identifying the required security attributes than the description of the security attributes provided in standard documentation format.

For identifying security patterns, our null hypothesis is

H02: ALSAF is no more useful in identifying suitable security patterns to be used in software architecture of an application than security patterns' documentation itself.

Our alternative hypothesis is:
H2: ALSAF is more useful in identifying suitable security patterns to be used in software architecture of an application than security patterns' documentation itself.

3.3 Experiment Design

We decided to use a Quasi-experimental design for this study as it had been anticipated that we would not have been able to assign the participants randomly to the control and treatment groups [23]. However, we knew that we would be able to manipulate independent variable, providing or not ALSAF. Moreover, we had also anticipated that we would not able to have a balanced design, i.e. the same number of subjects in each group (i.e., control and treatment) as we planned to recruit the practitioners from a training course. Hence, it was expected that many of them might decide to leave just after the training session. Table 1 shows the experimental design and it is clear that we were unable to anticipate the number of potential participants in each of the experimental condition.

TABLE 1: ALLOCATION OF THE PARTICIPANTS TO TREATMENT AND CONTROL CONDITIONS

Treatments / Material	ALSAF provided	ALSAF not provided
QVS	X number of individuals	X number of individuals

The **Independent variable** manipulated by this study is an architectural level security analysis framework, ALSAF, provided to the participants to support the task of identifying security sensitive quality attributes and potential patterns to satisfy those quality attributes when a list of security properties (i.e., general scenarios) have been provided with one treatment: ALSAF provided, and one control: ALSAF not provided.

The **Dependent variable** is the number of correctly identified security sensitive patterns required to satisfy the required security properties and the number of correctly identified security attributes from which the required security properties have been decomposed.

3.4 Participants

There are several sources of recruiting participants for an experiment such as practicing software engineers, students in postgraduate or undergraduate courses offered by tertiary educational institutes [24]. The participants can be recruited by offering some incentives or soliciting cooperation for a good cause. We recruited the participants for this study from a professional training course on software architecture evaluation offered by the author in Vietnam as part of a research project. We did not offer any incentive to the participants of this study. There were 24 participants in the course who were software engineering professionals employed in locally owned and multinational software development companies in Vietnam.

TABLE 2: PARTICIPANTS' LEVEL OF EXPERTISE IN SOFTWARE ARCHITECTURE RELATED KNOWLEDGE

Level of expertise	SA Design	SA Styles & patterns	Design patterns
None	0	2	3
Novice	12	13	12
Skilled	10	9	8
Expert	2	0	1

All of them had finished at least 4 yeas of university level IT related qualification. Their average work experience was approximately 5 years (61.7 months) with median of 60 months; and on average their age was 28 years. The positions held by the participants in their respective organizations ranged from software engineer to technical leader, project manager, and quality assurance manager. They possessed a good mix of skills in architecture design, implementation on different platforms using several kinds of object-oriented, procedural, and rule-based languages, software testing, maintenance, and technical support. Most of them had varying level of expertise in architecture design, architecture styles or patterns, and design patterns as shown in Table 2. Question on rating their level of expertise in the concepts and technologies used in the course has implications for the self-confidence of the participants. These data were collected at the beginning of the course.

3.5 Training

We have already mentioned that this study was incorporated in an industrial training course run by the author. This half day training course was designed to provide software engineering practitioners with an overview of modern concepts and practices of software architecture in general and software architecture evaluation in particular. The

course covered a broad range of topics[1] on software architecture evaluation, which have been summarized in [25]. During the seminar, the participants not only got engaged in discussions on the presented topics with reference to their own experiences but also performed small exercises on generating scenarios to specify quality requirements of a system of their choice and the use of suitable patterns to satisfy those scenarios. Hence, it is asserted that before the study, the participants had become familiar with the concepts of quality attributes and decomposing them into properties, general and concrete scenarios, design patterns, and design decisions. The training seminar used maintainability quality attribute for most of the examples and exercise. We did not introduce the security patterns and/or properties of ALSAF during the training in order to avoid any potential validity threat. However, we just encouraged the participants in the treatment group to try to use the framework to perform the required tasks. We placed this encouragement note on the experimental material provided to the treatment group.

3.6 Experimental Material and Logistics

3.6.1 Software requirements specifications

This study used the Software Requirement Specification (SRS) for a web-based portal called Qualification Verification System (QVS)[2]. The system was designed and developed to support the qualification verification activities of a recruitment process of a real company. The main objective of QVS was to make the recruitment process more reliable and foolproof by allowing potential recruiters to verify the educational and professional background and references of a job applicant. The system provides a number of features to support the different activities of the recruitment process: the candidate can register with system by providing the required information. The system provides automatic support to the verification officer to perform the verification and record keeping tasks. Once registered, candidates can perform a limited number of update tasks on their own records.

TABLE 3: SCENARIOS FOR SOME OF THE SECURITY REQUIREMENTS OF QVS

Series No.	Security Requirements
A	QVS provides online payments facility for the services that means all transactions to and from QVS must be secured.
B	QVS provides secured storage to customers' credit details and other information.
C	QVS should be able to identify and verify different users and their access privileges according to their memberships groups.
D	QVS shall be able to detect and prevent Denial Of Service (DOS) attacks and shall be able to run reliably most of the time.
E	QVS is an evolving system that shall easily be modifiable to introduce changes in the security policy and other security checks.

The system also enables a candidate and prospective employers to request the verification services, while a verification officer is able to use the system to perform verification requests made by candidates or prospective employers. Considering the nature of the business, security is one of the most important quality attributes (along with performance, maintainability and usability) of this system. We prepared a simplified version of the SRS, mainly related to security attribute, and a description of the system to provide the participants with as clear a picture of the system as possible. Table 4 shows some of the security requirements of QVS using scenarios, which has been described at an abstract level. Each of the scenarios can be used to characterise one or more security properties shown in the middle layer of ALSAF in Figure 1.

TABLE 4: FROM PROBLEM TO SOLUTION DOMAIN USING THE SECURITY FRAMEWORK

Series No.	Security Attributes	Security Properties	Patterns
A		Private communication/information protection - Defense in depth	
B		Data protection	
C		User identification, Access verification	
D		Reducing exposure to attack / Error prevention & handling	
E		Encapsulation of Security policy/ Initialization process	

3.6.2 Experimental task, support material, and marking scheme

The study required the participants to determine security sensitive pattern(s) (right layer of ALSAF) that can be expected to achieve a given set of security properties (middle layer of ALSAF). The participants were also required to identify those security attributes (left layer of ALSAF), which have been decomposed into security properties. That means the participants were required to establish relationships between security attributes, their respective

[1] Seminar slides are available from this URL - http://apsec2007.fuka.info.waseda.ac.jp/parts/SPL-ArchitectureEvaluation-Distribution.pdf

[2] System name is fictitious for maintaining the confidentiality but the objectives and described features of the system are correct.

properties, and the security patterns required to achieve those properties. The participants were expected to report their answers in the second and fourth columns of Table 4.

For support material, the control group was given a document that had textual description of the security attributes, properties, and patterns that we had distilled from security literature during the development of ALSAF as mentioned in Section 2. Detailed description of each elements can be found in [6]. The participants in the treatment group were provided with ALSAF as well as the document given to the control group. However, they were encouraged to use ALSAF framework for performing the experimental tasks.

We used a simple marking scheme. According to which each correctly identified security pattern(s) required to satisfy the required security properties received 1 mark. Same marking was applied for correctly identifying the security attributes; one mark for each correct answer. Based on the currently captured patterns and quality attributes in ALSAF, the maximum marks for patterns would be 14 and for quality attributes would be 13.

3.6.3 Post-experiment Questionnaire

All the participants in the study were asked to fill a questionnaire consisting of one question and some space for comments. The questionnaire asked the participant's to give their opinion of the usefulness of ALSAF for performing the task of identifying the security patterns and quality attributes for the given security properties on a five point scale: (1 = not useful at all, 2 = not very useful, 3 = useful, 4 = very useful, 5 = extremely useful).

3.7 Experimental Validity

3.7.1 *Threats to internal validity*

Internal validity is the degree to which the values of dependent variables can only be attributed to the experimental variables [26]. Since this study used a Quasi-experiment design, we are aware of the limitation of this kind of design especially the outcomes should be considered in terms of the strength of the causal statement.

Another threat to the internal validity of our experiment is the approach used to mark the outputs (i.e., identified attributes and patterns) of the study. One potential threat associated with the marking scheme is that the marker may assign incorrect scores to the solutions. We addressed this issue by having two markers independently mark the study's output. Both markers discussed and resolved the differences before finalizing the final score for each participant's output. Another mentionable threat to the internal validity can be the issue of researcher and subject expectations. We were evaluating a technology we ourselves developed and our enthusiasm for the technology could have influenced the participants and/or affected the way in which the answers were marked. The researcher tried to control this threat by seeking volunteers who did not have any relation with the researcher and by getting another researcher to be the primary marker of the participants' outputs. However, this issue can only be completely addressed if other researches are prepared to replicate this study.

3.7.2 *Threats to external validity*

External validity is the degree to which the results can be generalized [26]. In particular, it is important to consider whether or not the participants of this study can be representative of the software architects and designers who design and maintain software architectures in industry; another important point for consideration is whether or not the experimental materials and process used in this study are representatives of the process and materials used for designing software architectures of industrial applications. While the participants of this study were practitioners involved in software architecture related tasks in their daily activities, our results are most likely to apply to those software development professionals who have around 5 years of experience in software design and development. The requirement specifications used in the study were simplified version of a real system. The full system had a large set of functional and non-functional requirements, which would have required hundreds of large and small architectural decisions and considerations of applying patterns at various levels of abstraction. However, in industry, software designers usually have a much longer time period in which to understand problem, identify candidate patterns to address the problem and assessing their viability.

3.8 Experiment Conduct

The experiment was conducted as a part of a professional development training course on software architecture evaluation ran for software development practitioners in Vietnam. There were 24 attendees in the course, which ran for 4 hours with two breaks. At the beginning of the course, we announced the planned study and sought volunteers, who were willing to participate in the study after the course. Nineteen attendees of the training course

decided to stay for the study, which was planned to run for 45 minutes including 15 minutes for the debriefing session and getting post-experiment questionnaire filled.

After providing a brief description of the study and tasks to be performed, we distributed the material for the treatment and control groups to the participants arbitrarily (we can't claim randomization) based on the sitting arrangements in the training room. The participants were asked to read the requirements of the QVS including the scenarios that characterised the security requirements. Their task was to identify appropriate security sensitive patterns that could satisfy the security properties provided in Table 4 and describe which security attributes could be achieved using the identified patterns. They were asked to provide their answers in Table 4 by considering the security properties provided in that table. They were given 30 minutes to perform this task. They were encouraged to carefully read the support material provided to them as it may help them to perform the required task.

Once 30 minutes elapsed, the templates based on Table 4 filled with answers were collected from the participants. During the debriefing session, we introduce ALSAF to all the participants and explain to them its potential uses. Afterward they were asked to fill the post-experiment questionnaire to provide their opinions about the usefulness of ALSAF as mentioned in Section 3.5.3. Hence, all of the participants were asked to provide their views on that questionnaire irrespective of whether or not they had used ALSAF for the experimental tasks. We collected four sets of data: the demographic data, the identified quality attributes and patterns, and the questionnaire filled by all the participants to provide their opinions about the usefulness of ALSAF.

4. RESULTS AND ANALYSIS

The results of the descriptive statistical analysis of the raw data for identifying the security attributes based on a set of security properties are presented in Table 5. We decided to use a non-parametric test, Mann-Whitney, to compare the performance of the participants in the two experimental groups. We converted the data to ranks and performed one-tailed test on the ranked data. The results indicated that the average score for the treatment group was significantly larger than the average score for the control group (p=0.011).

TABLE 5: SUMMARY STATISTICS FOR IDENTIFYING SECURITY ATTRIBUTES

	Participants	Mean	Standard Deviation	Standard Error
ALSAF provided	9	4.56	1.24	0.412
ALSAF not provided	10	2.60	1.87	0.6

The results of the descriptive statistical analysis of the raw data for identifying the security patterns for a set of security properties are presented in Table 6. We again performed a non-parametric, Mann-Whitney, one-tailed test on the ranked data. The results indicated that the average score for the treatment group was significantly larger than the average score for the control group (p=0.022).

TABLE 6: SUMMARY STATISTICS FOR IDENTIFYING SECURITY PATTERNS

	Participants	Mean	Standard Deviation	Standard Error
ALSAF provided	9	5.78	3.2	1.1
ALSAF not provided	10	2.80	1.62	.512

The analysis of the answers to the post-experiment questionnaire revealed that a large majority of the respondents believed that ALSAF can be useful in identifying security attributes and patterns required for satisfying the identified security attributes. The responses from the participants were sought on a five point scale ranging from "not useful at all" to "extremely useful"; moreover, there was also space for explaining the choice. The frequencies of responses for each point on the scale are provided in the bracket along with each item on the scale: not useful at all (1), not very useful (1), useful (7), very useful (10), and extremely useful (0). There were also a few participants who provided some explanation for their choices on the scale. But the qualitative data were not enough to perform any analysis. The respondents who found ALSAF not useful did not provide any qualitative data.

5. SUMMARY AND CONCLUSION

The results of the analysis of quantitative data obtained by marking the solutions to tasks of identifying the required security quality attributes and patterns demonstrate significant effect of providing pattern-based security sensitive architectural knowledge captured in ALSAF compared with the standard documentation of security attributes, properties, and patterns. The treatment groups (i.e., participants provided with ALSAF) performed significantly better than the control groups (i.e., participants provided with standard documentation) for both identifying the quality attributes and identifying the patterns. The results of Man-Whitney statistical test to evaluate the hypotheses

for both experimental tasks revealed that the differences between the treatment and control groups for identifying security attributes and patterns were significant (p=0.011 and p=0.022 respectively). The findings from the statistical analysis indicate that ALSAF is more helpful in tasks involving identifying security attributes required in a system by looking at the properties (i.e., general scenarios) of that system and security patterns required to satisfy the security attributes compared with standard descriptive documentation of security attributes and patterns.

Apart from obtaining the objective quantitative data based on the solutions to the experimental tasks, we also gathered self-reported data by using one question with five point scale and space for open-ended comments. Analysis of the answers to the question about the usefulness of ALSAF revealed that a majority of the participants believed that ALSAF can be more useful during software architecture design and evaluation than the standard documentation of security attributes and patterns. Unfortunately, we did not get enough qualitative data explaining the reasons for selecting a particular choice on the scale.

It can be concluded that the results of our statistical analysis enable us to reject both of the null hypotheses in favour of the alternative hypotheses described in Section 3.2. Moreover, the subjective opinions of a large majority of the participants are consistent with the results of the statistical analysis. It should also be noted that the participants of the reported research did not get any training in using ALSAF for identifying security attributes and patterns. We are confident that a brief training on different ways (as outlined in Section 2.2) of using ALSAF during software architecture design and evaluation can significantly improve the results of using ALSAF.

When we proposed ALSAF as a mechanism to support architecture level security analysis in [6], we also delineated two short term goals to further this research:

- Develop a repository to store and access the architecturally sensitive security knowledge; and
- Assess the usefulness of the approach with experiments and case studies.

We have developed and trialled a repository of architectural knowledge to support the software architecture process as reported in [27, 28]. This paper has presented the design and results of one of the two experiments of a research project aimed at achieving the second short term goal. The research project purports to gather empirical evidence to assess the usefulness of ALSAF for systematically considering and addressing security related issues at the architecture level. The findings of the pilot and main study reported in this paper provide initial empirical evidence to support our claims about the usefulness of ALSAF. Hence, we can assert that we have successfully taken some concrete steps to achieve those short term goals. Now, while we are working on making it easy to identify and resolve conflicts between security and other quality attributes in the context of ALSAF as one of the long-term goals of our research, we hope researchers and practitioners will experiment with ALSAF to further validate it. In particular, we hope others will replicate our studies and contribute to a body of knowledge to determine the pros and cons of ALSAF to improve security sensitive architecture design and evaluation. We will also be happy to provide our tool for research purposes. We are also analysing the data from the second study of this research project. That study was also conducted with practitioners in another training course offered by the author. We are also working on reporting the lessons learned from conducting empirical studies during industrial training courses in general and in non-English speaking developing countries like Vietnam in particular.

6. ACKNOWLEDGMENTS

The researcher is grateful to the participants of this study and their organizations. Xiaowen Wang helped us to mark the participants' answers. Lero is funded by Science Foundation Ireland under grant number 03/CE2/I303-1.

7. REFERENCES

[1] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice. 2 ed. 2003: Addison-Wesley.
[2] CERT. CERT/CC Statistics 1988-2004. Last accessed on 26th February 2005, Available from: http://www.cert.org/stats/cert_stats.html.
[3] J. Viega and G. McGraw, Building Secure Software: How to Avoid Security Problems the Right Way. 2001: Addison-Wesley.
[4] A.v. Lamsweerde, Elaborating Security Requirements by Construction of Intentional Anti-Models, *Proc. of the 26th Int'l. Conf. on Software Eng. (ICSE)*, 2004.
[5] J. Yoder and J. Barcalow, Architectural Patterns for Enabling Application Security, *Proceedings of the 4th Pattern Languages of Programming*, 1997.
[6] M. Ali-Babar, X. Wang, and I. Gorton, Supporting Security Sensitive Architecture Design, *Proceedings of the First International Conference on Quality of Software Architectures (QoSA05)*, 2005.
[7] D.M. Kienzle and M.C. Elder. Final Technical Report: Security Patterns for Web Application Development. Last accessed on 28 March 2008, Available from: http://www.scrypt.net/~celer/securitypatterns/.
[8] M. Schumacher, Security Engineering with Patterns, in Lecture Notes in Compuer Science. 2003, Springer-Verlag GmbH.

[9] V.R. Basili, R.W. Selby, and D.H. Hutchens, Experimentation in Software Engineering, *IEEE Transactions on Software Engineering*, July, 1986. **12**(7): pp. 733-743.
[10] B.A. Kitchenham, et al., Preliminary guidelines for empirical research in software engineering, *IEEE Transactions on Software Engineering,* , 2002. **28**(8): pp. 721-734.
[11] I. Ozkaya, L. Bass, R. Nord, and R. Sangwan, Making Practical Use of Quality Attribute Information, *IEEE Software*, 2008. **March/April**: pp. 25-33.
[12] N. Lassing, P. Bengtsson, J. Bosch, and H.V. Vliet, Experience with ALMA: Architecture-Level Modifiability Analysis, *Journal of Systems and Software*, 2002. **61**(1): pp. 47-57.
[13] K. Petersson, T. Persson, and B.I. Sanden, Software Architecture as a Combination of Patterns, *CrossTalk The Journal of Defense Software Engineering*, Oct., 2003.
[14] F. Buschmann, Pattern-oriented software architecture : a system of patterns. 1996, Chichester ; New York: Wiley. xvi, 457 p.
[15] D.M. Kienzle, M.C. Elder, D. Tyree, and J. Edwards-Hewitt. Security Patterns Repository - Version 1.0. Last accessed on 28 March 2008, Available from: http://www.scrypt.net/~celer/securitypatterns/.
[16] M. Ali-Babar, B. Kitchenham, P. Maheshwari, and R. Jeffery, Mining Patterns for Improving Architecting Activities - A Research Program and Preliminary Assessment, *Proceedings of the 9th International conference on Empirical Assessment in Software Engineering*, 2005.
[17] M. Ali-Babar, B. Kitchenham, and P. Maheshwari, Assessing the Value of Architectural Information Extracted from Patterns for Architecting, *Proceedings of the 10th International conference on Empirical Assessment in Software Engineering*, 2006.
[18] M. Ali-Babar, B. Kitchenham, and P. Maheshwari, The Value of Architecturally Significant Information Extracted from Patterns: A Controlled Experiment, *Proceedings of the 17th Australian Software Engineering Conference*, 2006.
[19] L. Bass and B.E. John, Linking usability to software architecture patterns through general scenarios, *Journal of Systems and Software*, 2003. **66**(3): pp. 187-197.
[20] E. Folmer, J.v. Gurp, and J. Bosch, A Framework for Capturing the Relationship between Usability and Software Architecture, *Software Process Improvement and Practice*, 2003. **8**(2): pp. 67-87.
[21] L. Bass, M. Klein, and G. Moreno, Applicability of General Scenarios to the Architecture Tradeoff Analysis Method, *Tech Report CMU/SEI-2000-TR-014*, Softwar Engineering Institute, Carnegie Mellon University, 2001.
[22] C. Hofmeister, et al., A General Model of Software Architecture Design Derived from Five Industrial Approaches, *5th Working IEEE/IFIP Conference on Software Architecture (WICSA 05), Pittsburgh, PA, USA*, 2005.
[23] L.E. Toothaker and L. Miller, Introductory Statistics for the Behavioral Science. 1996, Pacific Grove, CA, USA: Brooks/Cole Publishing Company.
[24] M. Host, B. Regnell, and C. Wohlin, Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment, *Empirical Software Engineering*, 2000. **5**: pp. 201-214.
[25] M. Ali-Babar, Evaluating Product Line Architectures: Methods and Techniques, *The 14th Asia-Pacific Software Engineering Conference*, 2007.
[26] C. Wohlin, et al., Experimentation in Software Engineering: An Introduction. 2000: Kluwer Academic Publications.
[27] M. Ali-Babar and I. Gorton, A Tool for Managing Software Architecture Knowledge, *Proceedings of the 2nd Workshop on SHAring and Reusing architectural knowledge - Architecture, rationale, and Design Intent (SHARK/ADI 2007), Collocated with ICSE 2007.*, 2007.
[28] M. Ali-Babar, et al., Introducing Tool Support for Managing Architectural Knowledge: An Experience Report, *Proceedings of the 15th IEEE International Conference on Engineering Computer-Based Systems*, 2008.

Appendix A

The requirements specifications used for the study cannot fit within the limit of 10 pages that is why we have made it available from the following link:

http://www.staff.ul.ie/alibabar/qvsRequirements.pdf