An Exploratory Comparison of Security Patterns and Tactics to Harden Systems

René Noël[1], Gilberto Pedraza-García[2], Hernán Astudillo[3], Eduardo B. Fernández[3,4]

[1] Escuela de Ingeniería Civil Informática
Universidad de Valparaíso, Valparaíso, Chile
rene.noel@uv.cl

[2] Departamento de Sistemas y Computación
Universidad de los Andes, Bogotá, Colombia
g.pedraza56@uniandes.edu.co

[3] Departamento de Informática
Universidad Técnica Federico Santa María, Valparaíso, Chile
{hernan,edfernan}@inf.utfsm.cl

[4] (On leave from Florida Atlantic University, Boca Raton, FL, USA)

Abstract. The software architecture community considers non-functional requirements as key factors in designing a system architecture, and several approaches have been proposed to address them, including "architectural tactics". Specialized technical communities have developed approaches from their own perspective; in particular, security researchers have proposed "security patterns". This article describes a systematic attempt to compare both approaches, through an experimental study of the impact of chosen approach and participants' experience on the quality and effort of design decisions by non-security experts. We gathered practicing developers and graduate students, each group including novices and experts; trained subjects in both techniques; gave them a relatively simple problem (a tsunami warning system under current development); and measured the rate of effectively addressed threats (quality) and elapsed time to answer (effort). Based on previous experience, we had conjectured that security patterns would improve novices' quality but security tactics would improve experts' speed; however, preliminary results indicate that while experts were better than novices at identifying threats, they are no better at mitigating them. Further introspection suggests that more mature theories of tactics and patterns are still required for experimental comparison of architectural approaches.

Keywords: Architectural tactics, security patterns, secure architecture design.

1 Introduction

The architecture of a software system is designed by applying a set of decisions that define the main structure of components. Some of these decisions help to control the response of the quality attributes, other ensure achieving the functionality of the system. Design decisions about incorporating security in a software system have a

global effect; no local optimizations are possible. Being global, their full impact is only known when the system is completed.

Some of the most important types of decisions in software architecture are architecture patterns and tactics; thus, there are both patterns and tactics for systems security. We aimed to compare empirically relative performance of patterns and tactics techniques. In previous work, we have conducted extensive experimental validation of some software engineering results, for both agile methods [1], [2] and software architecture [3], [4]; we drawn on this experience to design a replicable experimental study with practicing developers and graduate students.

The document is organized as follows: Section 2 gives some background of tactics/patterns techniques, Section 3 presents the experimental design, Section 4 describes experiment operation details, Section 5 presents data and validity analysis, Section explores the results, and Section 7 summarizes and concludes.

2 Tactics and Patterns for Secure Systems

The building of secure software systems is a complex task because there are a great variety of threats [15]. One approach is to solve the security in the definition of software architecture. In the software architecture the most important decisions are taken about the functionality and quality properties of a software system [18]. Reusing knowledge allows to reduce risks (by preventing mistakes) and reduce costs (by providing readymade solutions). In particular, architecture knowledge (AK) has been characterized [8] as "design decisions + design", and it encompasses [14] not only decisions and rationale, but also [1] alternative solutions, significant entities from the problem space (e.g. stakeholders' concerns), technology constraints, business information, and general knowledge (e.g. design patterns); tactics, too.

Both patterns and tactics are strategies to reuse architectural knowledge. The secure software design community has proposed several techniques and mechanisms to reuse knowledge in building secure systems [5], using specialized design patterns to address identified threat and attack opportunities. The software architecture community has proposed using "tactics", with a higher abstraction and granularity level. However, more precise definition of these concepts is required, to support both architectural knowledge repositories [4] and secure design methodologies [15].

2.1 Security Architectural Patterns

Architecture patterns describe the high-level structure and behavior of software systems as solutions to multiple system requirements, whereas design patterns [18] focus on lower-, component-level decisions.

Design patterns are encapsulated solutions to recurrent problems in specific contexts; *security patterns* define solutions to handle threats or to fix a vulnerability [6]. Patterns are considered a good way to build secure systems and several methodologies based on them exist [6],[16]. Patterns include a solution to a security problem and several sections that define their use, applicability, advantages, and disadvantages.

2.2 Security Tactics

An alternative way to build secure systems is based on the idea of *tactics*, which are "measures" or "decisions" taken to improve some quality attribute concern [3],[4],[7]; they are "architectural building blocks from which architectural patterns are created" [2]. Architectural tactics are also seen as decisions that codify and record best practices for achieving some quality attribute [2].

A starting point for tactics systematization is the classification proposed together with the concept introduction [2]; later work [13] analyzed these tactics and related them to Common Criteria (standard security) requirements, but did not expand or refined the proposed classification, nor showed how to use them to build secure systems. We believe that, from a security viewpoint, this original list is inconsistent and incomplete, and we aim to enrich and expand it, at a detail level similar to the one already done for security design patterns.

A recent comparison of patterns and tactics (as a minor point in [7]) argued that *architecture patterns* provide scaffolding for incorporating *tactics* to the architecture, i.e. they have a subordinate relationship; however, *design patterns* and tactics are alternative decision tools, since both of them presume that key architecture decisions have been already taken. To our knowledge, patterns can be complementary and not alternatives because they can be used together: patterns can realize tactics.

3 Experiment Planning

3.1 Goal, Hypotheses and Variables

We propose to compare patterns and tactics with an experimental study, where both approaches were used by both novices (graduate students) and professionals, to harden a medium-size, realistic system design. Security patterns already have a well-document selection method and supporting catalog [5], which can be taught to novices to help identify patterns relevant to a specific circumstance; to enable direct comparison of the approaches' results, we developed a comparable catalog for security tactics based on Bass' [2]. To enable comparable evaluation of results from both approaches, we used annotations [7] over design documentation that was given to subjects when starting the experiment. Design documentation consists in sequence, component, deployment and domain UML diagrams.

We need to analyze the impact in the quality (rate of effectively addressed threats) and effort (elapsed time to answer) of design decisions of two independent variables: the technique used by the participants with two possible values: tactics and patterns and the experience of the participants with two possible values: graduated students with no experience in real software projects and professionals with experience making architectural design decisions. A 2x2 factorial experimental design allowed us to analyze both variables and its interactions. Also, we measured how many security threats the subjects identify. Table 1 shows four experimental groups; subjects are randomly assigned to a technique, considering their experience on making architectural decisions.

Table 1. 2x2 factorial design for comparing tactics and patterns.

	Novice Participants	**Expert Participants**
Technique 1: Tactics	Group 1	Group 3
Technique 2: Patterns	Group 2	Group 4

Based on the background presented, we had conjectured that security patterns would improve novices' quality and security tactics would improve experts' speed. To explore this conjecture, we stated the following hypotheses:

H0': There is no difference in Quality between groups.
H1': There is a difference in Quality between groups.

H0'': There is no difference in Effort between groups.
H1'': There is a difference in Effort between groups.

Also, we decided to add a threats identification step to the procedure, because we couldn't define if identification and solving of threats were separable mind processes, so no other hypothesis was stated:

H0''': There is no difference in the number of security threats identified between groups.
H1''': There is a difference in the number of security threats identified between groups.

In order to answer our research questions, we defined the following variables:

A. Dependent variables

- Elapsed time to answer (Effort). It measured the time that each participant spent in the tasks assigned.
- Rate of effectively addressed threats (Quality). Two pattern/tactic experts evaluate the treatment given by each participant to identified security threats.
- Number of threats identified by subjects (QoT). As explained later, subjects must identify security threats before applying tactics or patterns.

B. Independent variables

- Pattern group/ tactics group. Two different treatments were defined for the participants. One group was to focus on applying patterns in the architecture and the other group was to focus on applying tactics in the architecture.
- Experience/ novice participants. Participants are separated into two groups experts and novices.

3.2 Experimental Design

To test the hypotheses we designed a controlled experiment, and conducted a pilot application at Universidad Técnica Federico Santa María in August-September 2013.

1. *Participants:* The pilot study was done with students in a senior undergraduate class of software architecture and a group of practicing professionals with broad experience in software architecture definition.

2. *Study object:* For the comparative study, we used the architecture definition of a Tsunami alert system for the Chilean Coast. The architecture document describes the alert decision-making subsystem which is structured in a pattern of three layers: (1) the presentation layer contains components to interoperate with other systems and users; (2) the alert decision-making layer contains processing components for measuring seismic events, evaluation of information to begin the decision-making processes, alert level recommendation by region and generation of the official warning bulletin; (3) the data layer has components to persist seismic, sea level measurements, pre-modeling scenarios, decision support information, and warning bulletin. In addition, participants received the specification of a set of use cases, a description of sensitive resources, and security policies to identify attacks or threats and select patterns or tactics that best resolve and document architecture. This documentation was done with an ad-hoc notation.
3. *Blocking:* Training activities were performed to avoid confounding threats caused by previous knowledge of tactics or patterns in both types of subjects. Training covered concepts and application of both tactics and patterns, including the usage of catalogs. Before training, subjects were randomly assigned to Tactics and Patterns groups; a number is used to identify each participant to ensure the correspondence between the technique applied in training and in the experiment.
4. *Instrumentation:* The experiment was developed in three phases:
   A. *Training:* Consisted of a presentation of the concepts and principles for each technique, review of a catalog of patterns/tactics (respectively), a methodology for threats identification and applying patterns/tactics to solve them and exemplification of a notation to document the application of patterns/tactics in architecture. In addition, we conducted a training workshop to familiarize participants with the steps of the experimental procedure. In this workshop, the participant got to know and resolve doubts about the use of patterns/tactics catalog and annotations to describe the solutions. We propose these instruments and activities:
      - Application of a questionnaire containing participant information.
      - Training related to the applicability and use of security patterns / tactics in software architectures.
      - Training workshop on patterns / tactics.
      - Random selection of groups.
   B. *Experiment:* Participants were given an architecture description for a medium-sized, realistic system, and were asked to choose tactics or patterns from a catalog, in order to identify and mitigate security threats; the subjects were given the system description, the list of sensitive resources and security policies, and the tactics/patterns catalog. We proposed these activities and instruments:
      - Reading of the description of the problem.
      - Review of the use case document.
      - Understanding of the architecture of the tsunami warning system.
      - Analysis of sensitive resources and system security policies.
      - Identification of attacks or threats.
      - Selecting of patterns / tactics using the respective catalogs.

- Application of patterns / tactics in the architecture using the proposed notation.

C. *Post experiment:* We recorded the time needed to complete the activity, and the rationale behind the choices with a final report. We propose these instruments and activities:

- Application of the questionnaire with questions to gather information needed for interpretation and validation of the results
- Filling out the satisfaction survey.

The resulting architectures produced by each subject was evaluated by software architecture experts, who rated it by answering a set of questions about the quality of the decisions (i.e. consistence of selected tactics or patterns) and the effectiveness of the threat mitigation. Data analysis was focused on finding relationships between the independent variables, and the expert-given ratings of the architectures produced by the subjects.

4 Experiment Realization

4.1 Preparation

Two training sessions of two hours for both patterns and tactics were conducted before the experiment, for all novice participants in the experiment. In other hand, professionals received the same contents that were explained to novices during training: a presentation of the concepts and principles of patterns/tactics, a catalog of patterns/tactics, a method for applying each technique, and the notation to document the application of patterns/tactics in UML architecture documents. Also, a brief explaining session was made for clarification.

The training workshop was conducted with novice subjects; we tried to commit them by offering as inducement a few bonus points for final grade in the Software Architecture class for those who finished the activity; since this inducement was not linked to a performance evaluation, it encouraged participation but did not make subjects sensitive to their results. Each session was supervised by an experimenter, who did not interfere in the activity but delivered materials, solved instrumentation doubts and collected the forms at the end of the activity. Before the experiment we randomly distribute both novice and expert participants.

4.2 Execution

At the start of the experiment we deliver a guidance document with the activities to do. In addition to completing the guide activities, participants have information about the reference time to complete the activities.

We conducted the experimental activity in two sessions, one for the novice subjects and one for the experts, both under the same environmental conditions. In both sessions, subjects were randomly assigned to a technique (Patterns or Tactics). Table 2 shows the final distribution of subjects between groups.

Table 2. Final distribution of subjects between groups

	Novice Participants	**Expert Participants**
Technique 1: Tactics	6	2
Technique 2: Patterns	6	2

Previously prepared material was given to them, which consisted in the following documents:

1. *Experiment Guide:* A step-by-step guide of the activities involved in the experiment. The guide also presented several statements in order to allowing the subjects to make trade-off decisions. For instance, security policies, protected resources and protected operations were declared. These documents also allowed participants register the duration and results of each activity. The activities were presented in the following sequence:
   - Read the problem description.
   - Read the use case documentation.
   - Read the architecture document, including sequence, domain, component and deployment diagrams of the current solution.
   - Read Security priorities, including policies, sensitive resources and operations.
   - Identify security threats (an ad hoc form must be filled).
   - Select patterns or tactics to mitigate security threats (an ad hoc form must be filled) from the application catalog.
   - Apply patterns or tactics (modify UML design diagrams, by explaining the application of patterns or tactics through annotations over the diagrams).
2. *Problem documentation:* The problem description, use cases documentation and architectural design diagrams of the current solution.
3. *Support Documentation:* A tutorial for describing the results in UML diagrams, by using annotations as in [6], and a catalog of patterns or tactics, depending on the assigned technique.

4.3 Data Collection and Validation

Both experimental sessions (with inexperienced subjects and experts) were executed normally, and instrumentation seemed totally understandable, since participants did not raise any questions. After finishing the activity, which took between one and a half and two hours, a final form was completed by participants, in order to gather two types of information: rationale of their main architectural decisions, and a satisfaction survey in order to improve future experimental trials. All the documents were collected by the experiment driver, and no evidence of errors in executing the activities or filling the forms were found, although some resulting models were hard to read due to its complexity (and poor handwriting). Nevertheless, no data points were lost in this revision point.

Data collection continued with the experts' evaluation. The evaluation of the artifacts produced by the subjects had two major concerns: a large number of potential threats, and how to measure how well they mitigated these threats. To handle these

concerns, two experimenters, both experts in software architecture and security, described the key issues to identify and resolve threats and designed the review procedure. The evaluation design consisted in define a "ground truth" level, i.e. an expected "correct" output for the design decisions. This ground truth defined 13 basic threats (expected to be identified and mitigated by the subjects), and an example of a defense against the threat. A senior architect (Expert Evaluator) was asked to review the artifacts produced by the subjects and state if each threat was identified and how well it was mitigated, in a scale from 1 to 5. Table 3 shows the ground truth threats, and table 4 shows the evaluation scale for threats mitigation evaluation.

Table 3. Ground truth definition.

Use Case	Threat ID	Threat	Defense (expected)
Send Systemic Parameters	T11	Send of fake parameters	Digital sign
	T12	Illegal parameter reception	Expert authentication
	T13	Modification of received information	Authentication, authorization
	T14	DoS	Redundancy
Evaluate Systemic Information	T21	Modification of received information	Authentication, authorization
	T22	False warning generation	If insider: logger
			If responsible: logger + digital sign
			If expert: logger + digital sign
	T23	Decision is modified when sent	Encryption + digital sign
	T24	Warning is ignored by the responsible	Logger
Approve and send bulletin	T31	Bulletin is not sent	Logger
	T32	Bulletin is modified before being sent	Authentication, authorization, logger.
	T33	Bulletin is sent to a fake destination	Authentication
	T34	Bulletin is modified in transit	Encryption, digital sign
	T35	DoS	Redundancy

A third experimenter collected evaluations and validated that all subjects were evaluated by the expert evaluator. Finally, all experimenters reviewed the results; overall, it seemed like all subjects participated in the experiment seriously, and no data points were lost after this second checkpoint.

Table 4. Evaluation scale for expert's review

Evaluation scale	Result
5	Mitigates all cases in the threat
4	Mitigates almost all cases in the threat, but there are "hard exceptions"
3	Mitigates most of the normal cases, but a lot of cases still open
2	Mitigates trivial cases, but not the most
1	Threat is not mitigated

5 Analysis

A descriptive analysis was performed on the collected the data. We defined three variables to test the hypotheses: the quantity of threats detected by the subjects (QoT), the rate of effectively addressed threats (Quality), and the completion time for the activity, measured in minutes (Effort). We reduced data by calculating an average quality rate for each subject, among all the threats defined in the ground truth; e.g., a value of 3 in QoT for a subject means that he discovered 3 of the 13 threats of the ground truth; in other hand, a value of 4 in Quality, means that the average quality score for all the threats mitigated by the subject was 4 in a scale from 1 to 5.

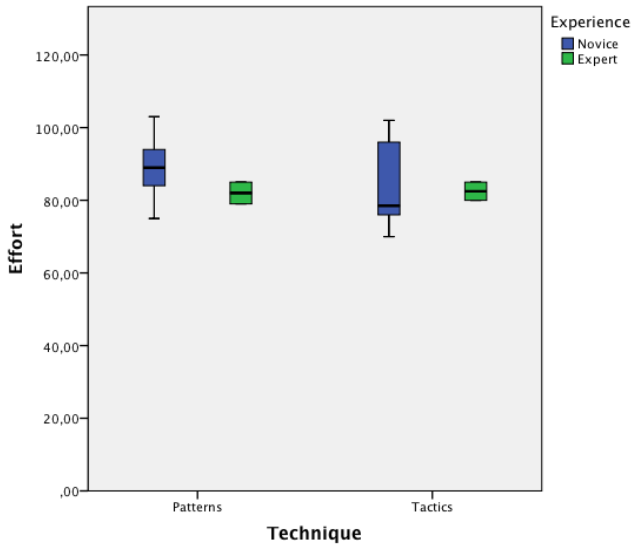

Fig. 1. Boxplot for Effort variable.

As seen in figure 2, two outliers were detected in the SPSS analysis, namely for the group Novice-Pattern and the variable QoT; however, this arises because all other data points have a value of 3, so we did not exclude this data points. No other outliers

were detected. Data points distribution can't to be assumed as normal for groups with n=2, i.e. experts groups. Non-parametric tests were applied for data analysis.

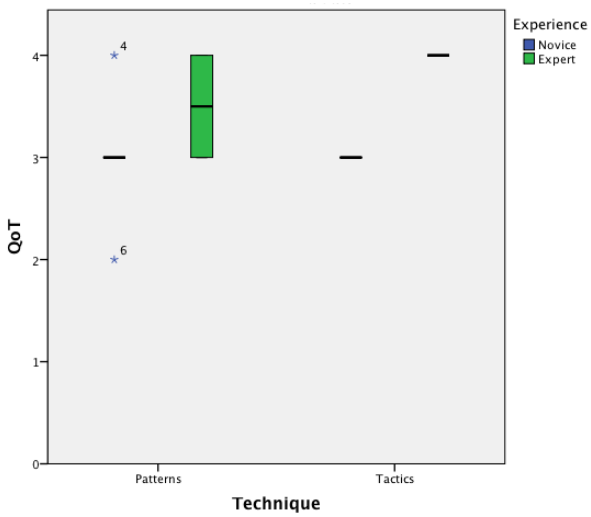

Fig. 2. Boxplot for QoT variable.

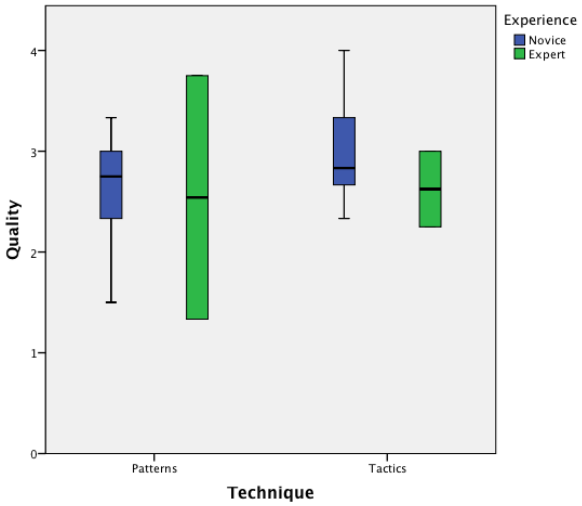

Fig. 3. Boxplot for Quality variable.

5.2 Descriptive Analysis

We first used descriptive statistics for the collected data to look for outliers and distributions for the variables Quality, Effort and QoT. Data was separated according to the four experimental groups. Figures 1, 2 and 3 show box plots for each variable.

5.3 Hypotheses Testing

The operational hypotheses were tested with Kruskall-Wallis H (KW) test and Median test, since parametric approaches like Two-Way ANOVA require normality assumption analysis but the low quantity of subjects in experts groups (2 subjects for each one) couldn't ensure this condition. We performed the tests for Effort, QoT and Quality variables. To determine statistically significant differences among groups, we took four combinations of experience and technique as separated groups. Table 5 shows the medians for all three variables.

Table 5. Medians for Effort, QoT and Quality.

Group	QoT	Quality	Effort
Novice-Pattern	3,00	2,75	89,00
Novice-Tactic	3,00	2,83	78,50
Expert-Pattern	3,50	2,54	82,00
Expert-Tactic	4,00	2,63	82,50
Total	3,00	2,83	84,50

For Quality, there were no statistically significant differences between four groups (p=0,696 for KW). Also, Medians are not different between groups (p=0,721 for Median test). Figure 4 shows the Medians between groups.

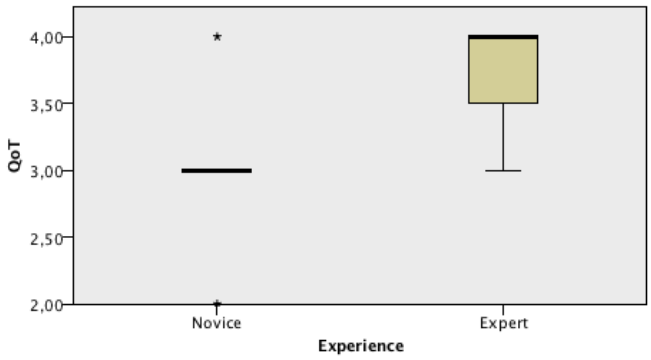

Fig. 4. Boxplot for Experience groups

For QoT, KW no significant differences can be assumed, although *p* value is close to 0,05 (p=0.072). In other hand, Median test did show significant median differences between groups (p=0.31).

For Quality, no differences between groups were found for both tests (p=1 for KW, p=0.81 for median test).

We analyzed the data considering technique as the only independent variable, and no significant differences were found in distribution or median for Effort (p=0,371 in KW, p=619 in median test), QoT (p=0.699 in KW, p=1 in median test), and Quality (p=0.596 for KW, p=1 for Median).

Finally, we grouped data only considering experience, and statistically significant results show that there are differences in QoT for distribution (p=0.014) and medians (0.027). Figure 4 and table 5 describe medians for two groups.

Table 5. Medians for Experience groups.

Experience	QoT	Quality	Effort
Novice	3,00	2,83	85,00
Expert	4,00	2,63	82,50
Total	3,00	2,83	84,50

Descriptive statistics for data grouped by experience show that there are no outliers in any groups.

Finally, we can state that experts did identify more threats than novices, but there were no differences in the quality of the mitigation actions nor in the effort needed to mitigate them.

5.4 Validity Analysis

Considering that the experiment focus is theory testing (as per [17]), two main validity types are priority: internal validity, and construct validity. One of the main threats that we tried to mitigate was the construct invalidity due to the lack of definition of Tactics theory. Tactics are global decisions that affect the system as a whole, while patterns are more specific to a particular situation expressed in use cases. This could lead to misjudging the answers given by the participants, and it forced us to raise questions about the procedure to apply a tactic, and how its results can be clearly represented and objectively assessed. Thus, we defined a procedure that could be similar and comparable with Patterns: threat mitigation. The input artifacts for applying Tactics are not clear in the theory, so we wondered whether the problem and the system design model were enough input for applying the technique or we needed to provide a list of threats to be mitigated with Tactics. We decided to add a threat identification step to the procedure, because we couldn't decide whether threat identification and mitigation are separable mind processes. As results show, threat identification can be an important issue: experts identified more threats, but the final average score for threat mitigation was not statistically significant compared with novices. This led us to ask if an average score is a fair quality metric: experts and novices have the same rate of quality, but experts mitigated more threats. But are all threats equivalent in complexity or in impact over the system security? How can we weigh the scores by the mitigated threat? In our design, quality, in terms of threat mitigation, was assessed by a senior architect, but having more evaluators would improve the validity of our experimental procedure.

Internal validity can be controlled by experimental design. Our main concern was previous knowledge about applicability of tactics and patterns. Using students as novices, the threat was that they did not have prior knowledge about patterns or tactics. We solved it by leveling subjects with training activities and pilot exercises

that all the subjects completed with no problems. Factorial design, and single trial and treatment for each subject helped us to strengthen this type of validity.

6 Result Analysis

Beyond hypotheses testing, the experimental design process led us to question the theoretical basis of each technique. Thus, we looked for conceptual and procedural knowledge of the tactics approach to design the activity according to bibliographical referents, and we found that tactics definitions are rather vague (e.g. "add confidentiality"), and interpreting them required some domain knowledge (e.g. "using a firewall" requires to know what a firewall is, how it fits into an otherwise normal architecture, and what its implications are on the system installation and configuration). Moreover, the current list of security tactics (as shown in [2] and others) is incomplete: some are missing (like information hiding), some are actually design principles (like "limit exposure"), and some are redundant. From a secure systems perspective, the current list of tactics seems to confuse the "what" with the "how"; these are not inherent flaws of tactics themselves, but point to the need for a careful systematization and classification review.

7 Conclusions

We conducted a 2x2 factorial experiment with novices and experts in order to compare two techniques for harden systems. We designed a step-by-step problem solving activity to explore differences in quality and effort for architectural tactics and security patterns. We didn't find statistically significant differences in these variables, but found that expert subjects were able to identify more security threats than novices. This makes us think that threat identification is a key issue in improving systems security, independently of technique to help novices to be as effective and efficient as experts, and this activity is a key issue to be covered by techniques that look for improving systems security.

Acknowledgments. This work was partially supported by UTFSM (grant DGIP 24.12.50, CCTVal Basal FB0821), and AGCI Chile Student Mobility Platform of the Pacific Alliance (PhD research internship 2013).

References

1. Avgeriou, P., Kruchten, P., Lago, P., Grisham, P., Perry, D.: "Architectural knowledge and rationale: Issues, trends, challenges." SIGSOFT Softw. Eng. Notes 32(4), 41-46 (Jul 2007).
2. Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice,* 3rd ed. Addison-Wesley Professional (2012).
3. Braz, F., Fernandez, E.B., and VanHilst, M.: "Eliciting security requirements through misuse activities". Procs of the 2nd Int. Workshop on Secure Systems Methodologies using Patterns

(SPattern'08). In conjunction with the 4th International Conference on Trust, Privacy & Security in Digital Business (TrustBus'08), Turin, Italy, Sept 1-5, 2008. 328-333.

4. Capilla, R., Zimmermann, O., Zdun, W., Avgeriou, P., Küster, J.M.: "An Enhanced Architectural Knowledge Metamodel Linking Architectural Design Decisions to other Artifacts in the Software Engineering Lifecycle." In Proceedings of 5th European Conference on Software Architecture, ECSA 2011, pp. 303–318 Springer (2011).
5. Fernandez, E. B., Astudillo, H.: "Should we use tactics or patterns to build secure systems?" In: First International Symposium on Software Architecture and Patterns, in conjunction with the 10th Latin American and Caribbean Conference for Engineering and Technology. pp. 23–27 (2012).
6. Fernandez, E. B., VanHilst, M., Larrondo Petrie, M., Huang, S.: "Defining security requirements through misuse actions." In: Ochoa, S., Roman, G.C. (eds.) Advanced Software Engineering: Expanding the Frontiers of Software Technology, IFIP International Federation for Information Processing, vol. 219, pp. 123-137. Springer US (2006).
7. Harrison, N.B., Avgeriou, P.: "How do architecture patterns and tactics interact? A model and annotation." J. Syst. Softw. 83(10), 1735-1758 (Oct 2010).
8. Kruchten, P., Lago, P., van Vliet, H.: "Building up and reasoning about architectural knowledge." In: Hofmeister, C., Crnkovic, I., Reussner, R. (eds.) Quality of Software Architectures, Lecture Notes in Computer Science, vol. 4214, pp. 43-58. Springer (2006).
9. López, C., Codocedo, V., Astudillo, H., Cysneiros, L.M.: "Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach." Sci. Comput. Program. 77(1), 66-80 (Jan 2012).
10. López, C., Inostroza, P., Cysneiros, L.M., Astudillo, H.: "Visualization and comparison of architecture rationale with semantic web technologies." J. Syst. Softw. 82(8), 1198-1210 (Aug 2009).
11. Nöel, R., Valdes, G., Visconti, M., Astudillo, H.: "Adding planned design to XP might help novices' productivity (or might not): Two controlled experiments." In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 285-287. ESEM '08, ACM, New York, NY, USA (2008).
12. Nöel, R., Valdes, G., Visconti, M., Astudillo, H.: "Deconstructing agile processes: Would planned design be helpful in xp projects?" In: Proceedings of the 2008 International Conference of the Chilean Computer Science Society. pp. 42-51. SCCC '08, IEEE Computer Society, Washington, DC, USA (2008).
13. Preschern, C.: "Catalog of Security Tactics linked to Common Criteria Requirements." Proceedings of Pattern Languages of Programs (2012).
14. Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Ali Babar, M.: "A comparative study of architecture knowledge management tools." J. Syst. Softw. 83(3), 352-370 (Mar 2010).
15. Uzunov, A.V., Fernandez, E.B., Falkner, K.: "Securing distributed systems using patterns: A survey." Computers & Security 31(5), 681-703 (2012).
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Longman Publishing, Boston, MA, USA (1995).
17. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: "Analysis and interpretation." In: *Experimentation in Software Engineering*, pp. 123-151.Springer Berlin Heidelberg (2012).
18. Lago, P., Avgeriou, P., Kruchten, P.: "Organizing a software architecture body of knowledge: summary of the 5th SHARK workshop," at ICSE 2010. ACM SIGSOFT Software Engineering Notes 35(5), 37-40 (2010)