Does Organizing Security Patterns Focus Architectural Choices?

Koen Yskout
IBBT-DistriNet
KU Leuven
Heverlee, Belgium

Riccardo Scandariato
IBBT-DistriNet
KU Leuven
Heverlee, Belgium

Wouter Joosen
IBBT-DistriNet
KU Leuven
Heverlee, Belgium

Abstract—Security patterns can be a valuable vehicle to design secure software. Several proposals have been advanced to improve the usability of security patterns. They often describe extra annotations to be included in the pattern documentation. This paper presents an empirical study that validates whether those proposals provide any real benefit for software architects. A controlled experiment has been executed with 90 master students, who have performed several design tasks involving the hardening of a software architecture via security patterns. The results show that annotations produce benefits in terms of a reduced number of alternatives that need to be considered during the selection of a suitable pattern. However, they do not reduce the time spent in the selection process.

Keywords—secure software engineering; experiment; security patterns; software architecture

I. INTRODUCTION

A security pattern describes a particular recurring security problem that arises in specific contexts and presents a generic scheme for its solution [1]. The description of a pattern follows the familiar template of, for instance, the Gang of Four patterns, including sections on the problem, the forces, the solution, the known uses, and so on. Security patterns can be a valuable vehicle to design secure software, as they provide sound, time-proven solutions. An inventory done in 2007 counted over 220 security patterns that had been documented over the previous ten years [2]. While there is clearly no shortage of security patterns in the literature, they are not nearly as popular as the software design patterns (e.g., Gang of Four). An investigation by Laverdiere et al. [3] has discovered that security patterns are plagued by a number of issues, including over- and under-specification, lack of generality and consensus, as well as misrepresentation. The mass of security patterns has also led to overlaps, as reported by Hafiz et al. [4]. These are all potential setbacks for a wider adoption.

The secure software engineering community is trying to mitigate the present situation by focusing on the rationalization of the patterns landscape, in order to make the offering more accessible, structured, and effective. In this stream, most activities have focused on (1) classifying patterns, e.g., via taxonomies and (2) provide support for navigation among patterns that are related, e.g., via pattern languages. These proposals result into extra tags and annotations to be included in the pattern documentation in order to improve

usability [5]. However, to date, no evidence has been collected that any of these annotations do provide an advantage to the end users, i.e., architects and software designers.

This paper contributes with a pioneering empirical study in the domain of security patterns. In particular, this work focuses on security patterns for software architecture design as well as detailed design. The key research question is *whether security patterns annotations increase the performance of an architect when making architectural choices*. In the context of this work, an architectural choice amounts to the evaluation and selection of a solution in the form of a security pattern.

For the study, we use a specific proposal of 4 annotation types that has been developed at KU Leuven for teaching purposes and has been applied to a catalog of 35 security patterns [6]. The annotations provide extra information about the applicability of each pattern, its relationships with high-level security goals and other patterns, and its impact on other software qualities.

In summary, the study divided 45 teams of master students in two treatment groups. The students had to perform 4 design tasks involving the hardening of a software architecture via security patterns. One treatment group had access to a version of the catalog that is augmented with the above-mentioned annotations. The other group had access to the plain pattern documentation only. We have measured the performance of the teams in terms of (1) the time it takes to carry out each task (as an approximation of effort) and (2) efficiency, i.e., the number of patterns that are browsed in order to come to a solution. The results suggest that there is no significant difference between the two groups in terms of time. However, the group using the annotated catalog is more efficient. At the end of the experiment a questionnaire was also administered in order to validate the experimental assumptions and gather feedback from the participants.

The rest of this paper is structured as follows. The annotation scheme used in the experiment is described in Section II and positioned with respect to the related work in Section III. The design of the experiment is described in Section IV, while the results are presented in Section V. The data obtained via the follow-up questionnaire are explained in Section VI. Finally, Section VII discusses the threats to validity and Section VIII gives the concluding remarks.

II. ANNOTATIONS

While pattern descriptions are expected to contain sufficient contextual information to assist with selecting a suitable pattern, this is not always the case with security patterns. As mentioned before, annotations have been proposed to provide this information. The study takes the perspective of the software architect, and aims at measuring the effect of such annotations on the architect's performance.

For the experiment, the annotations described in [6] are used. This choice was subject to two criteria. Firstly, the annotations used in this study have to be representative of what has been suggested in the state-of-the-art. The selected set is a combination of existing proposals, as illustrated in the related work section. Therefore, it is expected that the annotations of other proposals would yield to similar results, although this should be verified in a separate experiment. Secondly, a catalog needs to be available in which the annotations have been applied to actual patterns. The catalog used in this study already existed (including the annotations) and has been tested for over three years in courses with students. In the past, the students have used the catalog to build realistically sized systems and have provided positive feedback: it covers the solution space well and is manageable in a lab setup.

The annotations in the catalog cover four dimensions:

- the *security objective(s)* for which the pattern provides a solution, for example confidential data transmission, data storage integrity, accountability and so on;
- the *applicability* of the pattern to either the high-level architecture of a system or its detailed design. This annotation also indicates whether the pattern is to be applied in the core of the system or rather in its deployment environment;
- the *trade-off labels*, which indicate the positive or negative impact of the pattern on other software qualities, such as performance or maintainability;
- the *relationships* among patterns, e.g., functional dependency, mutually-exclusive conflict, alternative, and so on.

For example, the annotations for the SECURE LOGGER pattern determine that it is linked to the 'auditability' objective, is applicable to the core of a system architecture, impacts positively on maintainability and negatively on performance, and is related to the SECURE PIPE and AUDIT INTERCEPTOR patterns, which are advised to be used as supporting solutions [7].

III. RELATED WORK

An extensive survey on security patterns has been published by Yoshioka et al. [8] and two referential books are available [9], [1]. To the best of the authors' knowledge, there are no empirical studies concerning security patterns.

Over the last decade, various proposals have appeared for organizing and classifying the security patterns. In this

section we illustrate how the proposal used in this study and summarized in Section II is indeed connected to (and representative of) the state-of-the-art.

Security objectives. Hafiz et al. [5] survey the taxonomies that have been presented so far and propose a new schema based on the following dimensions: (1) the addressed security issues (i.e., CIA—confidentiality, integrity, availability), (2) the application context (i.e., core security, perimeter security and exterior security), (3) the Zachman framework (stakeholders vs. concerns [10]), and (4) the Microsoft STRIDE threat categories [11]. The proposal used in this study uses the security objectives as one of the classification dimensions. The security objectives are, in fact, a refinement of the CIA. Security objectives are meant to provide a connect between the solution space and the problem domain. In this perspective, Rosado et al. [12] map out the relationships between security requirements and security patterns. Mazhelis and Naumenko [13] suggest a similar approach.

Applicability. The proposal used in this study includes the applicability of a pattern, which is similar to the application context (e.g., core security vs perimeter security) in the work of Hafiz et al [5]. The applicability also marks a distinction between high-level architectural design and detailed design. This is in line with the work of Rosado et al. [12], proposing the same differentiation, and similar to the works of Yoshioka et al. [8] and VanHilst et al. [14], classifying security patterns according to the software life-cycle phases (such as requirements, architecture and design, implementation).

Relationships. Hafiz et al. [5] acknowledge that classification is only one side of organization. As important as classification is navigation; that is, the ability of guiding the reader toward the selection of a pattern. Those authors point out that the relationships among patterns represent a key ingredient to successful navigation. The relationships used in this study are similar to the dependencies among security problem patterns suggested by Hatebur et al. [15]. An interesting work by Fernandez et al. [16] outlines a methodology to automatically elicit the dependencies via the analysis of the textual description of the patterns.

Trade-offs labels. Concerning the trade-off labels, they are similar to the work of Weiss [17], which traces the contribution (either positive or negative) of security patterns to qualities like availability, performance, cost, maintainability, usability, and the CIA itself.

IV. EXPERIMENT DESIGN

The experiment has been performed with master students as part of the lab project for a course on software architecture. The experiment ran over a period of two weeks and was mainly executed during two supervised lab sessions (5 hours in total), although the students were given the option of continuing working on the project at home (while still being monitored by means of the tool support, as described later). The students worked in teams of two people and we

have measured the performance of each team as a whole, as described later on. The teams have performed five design tasks, each of which consisted of hardening an existing software architecture with respect to a number of security requirements. To assist the teams in accomplishing these tasks, the tool they used provided access to a catalog of security patterns. The catalog contains 35 security patterns, mostly coming from a referential book [9]. Two versions of the same catalog have been used. One version (called PLAIN) contains the pattern documentation as it is presented in the literature. In the other version (called ANNOTATED), the patterns are enriched with the annotations mentioned in Section II. Accordingly, the teams have been divided in two balanced groups (called the PLAIN and ANNOTATED group from now on) depending on the version of the catalog they were provided with. Per design task, the average performance of the two groups has been compared.

We are interested in determining if the presence of annotations improved the performance of the teams regarding the selection of a suitable solution from the security pattern catalog, and if so, to what extent. For this, we interpret performance in two ways. First, we look at the *selection time*, i.e., the time that each team spent on selecting one or more patterns from the catalog. The expected outcome is that the teams in the ANNOTATED group will arrive at their selection more rapidly, because they can use the annotations to quickly discard irrelevant patterns and thus work faster. Second, we investigate the *selection efficiency* (or efficiency for short) of each team, namely whether the teams actually discarded the irrelevant patterns, and only looked at the patterns they eventually used for solving the task. Again, it is expected that the presence of the annotations makes the ANNOTATED group more efficient.

The rest of this section further describes the context, preparation and setup of the experiment, and the precise measurements and research hypotheses. For more details about the experiment's material and setup, we refer to this paper's accompanying website [7].

A. Experimental Object

The system that was used for the experiment comes from the domain of electronic health care. It allows a cardiologist to remotely monitor patients with cardiovascular diseases. The system consists of a central Patient Monitoring System (PMS) with which various external entities communicate. The data about a patient is obtained from sensors, for example a wearable unit that continuously measures the heart rate and blood pressure of the patient. These sensor readings are collected and sent to the PMS by a gateway device (e.g., a smartphone). The system stores the received data in a sensor reading database, analyzes it, and alerts the patient's cardiologist if further action needs to be taken. Also, when the readings indicate an emergency (e.g., an imminent heart attack), the emergency services are notified.

Table I
BACKGROUND OF THE PARTICIPANTS

Program & specializations	PLAIN	ANNOTATED	Total
Computer Science	31	33	64
Distributed Systems	5	5	10
Secure Software	8	8	16
Software Engineering	3	2	5
(Other)	15	18	33
Informatics	4	6	10
Applied Informatics	7	6	13
Other	2	1	3
TOTAL	44	46	90

Further, the system is also accessible by people from the patient's close environment (e.g., friends, the patient's general practitioner, or home caretakers). They can review the patient's status, and provide additional information by filling out questionnaires. The initial setup and configuration of the system for a patient is done by a nurse that has been trained for this purpose. Finally, the system should interact with the Hospital Information System (HIS) of the hospital for exchanging data and/or requesting emergency assistance.

B. Participants and Teams

The experiment has been executed with 90 students in their first year of a two-year master in computer science at the KU Leuven. The students have different backgrounds, for example secure software, artificial intelligence, or human-computer interaction. They were enrolled in an obligatory course on software architecture and the work done by the students in the context of the experiment was integral part of the course syllabus. As mentioned before, the participants worked in pairs and have been allowed to choose their own teammate. This is a standard practice at KU Leuven and the experimenters could not control that (e.g., via random team assignment). The 45 teams have been randomly assigned to the two treatment groups, leading to 22 teams in the PLAIN group and 23 teams in the ANNOTATED group. By using stratified randomization, an attempt has been made to balance the assignment of teams to groups with respect to the background of the students. In Table I, the distribution of the participating students across the different study programs, specializations and groups are given. Note that sometimes, within a single team, each member had a different background. This has hindered a perfectly balanced group assignment. However, participants with a security background are evenly distributed across the two treatment groups.

The students were made aware of their participation to an experiment and of the fact that their work was monitored, although the goals (hypotheses) of the experiment were not communicated to them. It has been made clear to the participants that the experimenters would not look at the raw measurements until after their grades for the course were made official. This pledge is important because the course

was taught by the experimenters themselves and we wanted to avoid any perception of coercion from the participants. The message has been reinforced several times during the duration of the course. Furthermore, the participants were given a ten days grace period to opt out from the experiment after the course's grades were made official and before the experimenters started analyzing the measurements. No participant decided to opt out.

C. Course and Lab Material

The experiment was part of the hands-on project for a one-semester course on software architecture, which consisted of two parts. In the first part (4 weeks), the students were taught to build a software architecture, following the attribute-driven design method [18]. In the accompanying lab project, the students created and documented (in UML) the architecture for the PMS system outlined before. This part of the project was not part of the experiment.

During the second part of the course, the students have received four lectures on various security-related topics such as understanding, eliciting and documenting security requirements, documenting security architectures, and using security patterns. The students were also given a tutorial about the security patterns contained in the catalog. The students in the ANNOTATED group received an additional (very short) lecture, explaining the annotations that were available to them in the catalog. In the accompanying lab project, an initial architecture of the PMS system (the same for all teams) needed to be hardened by instantiating security patterns. The architecture was created by the experiment designers and explained to the participants beforehand. The documentation of the PMS architecture is not detailed here due to space limitations and is provided to the interested reader on the website [7].

During the first part of the course, the students were free to choose the tools to work with, e.g., for drawing UML diagrams. To facilitate the collection of the measurements for the experiment, during the second part the students have been obliged to use a custom tool that we provided (and that was unavailable during the first part). Before the experiment started, the participants have received a tutorial and a demonstration on how to use the tool. All lab material (including the initial architecture, the catalog and the tasks to be performed) were digitally provided to the participants by means of the tool. Copies of the lab material are available on the website. More details about the tool are given later in Section IV-F.

D. Tasks

Each task consists of elaborating the software architecture in order to support a security requirement. The requirement is documented via a description of the threats to protect against, in the form of two misuse cases [19], and a description of how the system is expected to react, in the

form of a quality attribute scenario [18]. Table II contains a high-level description of the five tasks that were given to the teams. The table also contains the expected impact of the task on the architecture and the patterns from the catalog that could possibly be used for completing the task. Clearly, this additional information was not provided to the students. Note also that, in tasks A and B, the correct solution requires the selection of a system of collaborating patterns, while in tasks C–E a single pattern can be found that solves the problem entirely. Further, task E is expected to be the easiest one to solve.

E. Process

All teams were given a process to follow when hardening the architecture. This provides the students with the necessary guidance so that they do not get “stuck” with the tasks, as they never used security patterns before. The process is lightweight and very intuitive, as it resembles any problem solving approach: a list of potential solutions is scouted first and then the most adequate is chosen. For each task, the teams follow the steps below:

- 1) *Study* the requirement that should be implemented and assess which parts of the architecture are impacted and how. If the requirement is already sufficiently supported by the current architecture, skip all remaining steps and go to the next task.
- 2) Quickly skim through the security pattern catalog, and create a *shortlist* of possibly interesting patterns for the current task.
- 3) Study the patterns from the shortlist more thoroughly, evaluate the real effectiveness of each candidate, make trade-offs with other qualities that are important for the architecture, detect conflicts with already implemented patterns and make a *final selection* of patterns to instantiate.
- 4) *Instantiate* the solution in the architecture by injecting the pattern in the design. This step is not used in the context of this experiment. The observations on this step are part of a follow-up study.

Note that participants were not requested to build a secure architecture from the ground up. Therefore, there was no need for them to follow a more articulated architecture creation process like the Attribute Driven Design [18].

For the teams in the ANNOTATED group, the process steps are the same, but the activities are augmented so that the students can take advantage of the annotations in the catalog they are using. In particular:

- In step 1, the team must assign a security objective to the requirement.
- In step 2, the security objective is to be used to quickly select the relevant patterns for the shortlist. Further, the annotations about the relationships with other patterns can be used to extend the shortlist.

Table II
TASK DESCRIPTIONS AND IMPACT

Task	Scenario to prevent	Security objective and expected impact	Possible pattern(s)
Task A (warm-up)	A malicious person attempts to bring down the sensor reading database. This leads to the loss of received readings, and results in the unavailability of up-to-date information for the cardiologist.	AVAILABILITY. The sensor reading database is replicated, possibly using different database technologies. The sensor reading database host is shielded from the external network. Difficulty: hard	Firewall; Load Balancer; Replicated System
Task B	A user of the system tries to avoid responsibility for his action, by tampering with the log files.	ACCOUNTABILITY. Only authenticated users can perform actions, and they are logged in a tamper-proof (append-only) audit log. Difficulty: hard	Authentication Enforcer; Audit Interceptor; Secure Logger
Task C	Unauthorized entities attempt to access sensor data about a patient (which is sensitive medical data).	CONFIDENTIAL DATA TRANSMISSION. Data in transit between the patient's gateway and the PMS system is encrypted, for example using SSL. Difficulty: average	Secure Pipe
Task D	A member of a patient's personal health community tries to use the web interface to access information about a patient to which she is not related.	CONFIDENTIALITY. All actions that are performed by members of the community are checked against a policy. All necessary information to make an authorization decision is available to the component that makes the decision. Difficulty: average	Authorization Enforcer
Task E	The web interface of the personal health community is targeted by a malicious entity, by providing malicious input in order to cause system malfunctioning or unauthorized access to patient information.	DATA INTEGRITY. The system is designed so that external input is validated before it is used. Difficulty: easy	Input Guard

- In step 3, the annotations about the conflicts with other patterns and the trade-offs (with performance, availability and modifiability) are to be used to facilitate the final selection.

F. Experiment Execution

The process is enforced by a tool, in order to ensure that it was followed closely by each team. The use of tool is important to counter a common threat to internal validity, i.e., when the participants do not follow the planned lab procedures. The tool is a customized Eclipse environment, consisting of the Topcased UML Editor (topcased.org), a viewer to display the description of the tasks to be performed, a browser to navigate the security pattern catalog, and a wizard to enforce the process described above.

The first task (Task A in Table II) is a warm-up task in order for the team to become familiar with the system, the tool and the pattern catalog. Each team performs this task first. The other four tasks (B, C, D, and E) have to be completed sequentially (i.e., postponing a task is not allowed), and incrementally (i.e., each task is executed in the context of the architecture resulting from the previous tasks). In order to weed out the learning effect, the order of execution of tasks B–E is randomized across the teams. The tasks are provided to the teams via the tool (i.e., not on paper or online) to make it harder to pass the task descriptions around. Similarly, the security patterns catalog is only available through the tool, and could not be printed. Each team receives a unique code, which is entered when first starting the tool. The tool then configures itself for that specific team, for example by loading the correct order of the tasks and enabling or disabling the display of the annotations

in the catalog browser, depending on the team's assignment to the treatment groups.

For the PLAIN group, the browser to navigate the catalog is a simple HTML viewer with an index of all available patterns. For the ANNOTATED group, the browser allows the filtering of the index according to a specific security objective, the applicability of the pattern, the trade-off labels of interest, or a combination of the above.

As mentioned before, a wizard enforces the strictly sequential execution of the tasks, and of the process steps within each task. For instance, during step 1 (study of the requirement), the team has to use the wizard to enter its considerations about the impact of the requirement and, for the ANNOTATED group, the security objective that has been assigned to the requirement. In steps 2 (shortlisting) and 3 (selection), the wizard must be used to enter the shortlist and the final selection. During steps 1–3, the diagrams of the architecture are accessible in read-only mode via the UML editor. The UML tab is enabled for editing during step 4 only (instantiation of the pattern). During the lab sessions, one or more supervisors were always present to answer questions about the functionality of the tool, the meaning of the tasks and so on.

The tool is instrumented to monitor the team's actions in the background. In particular, the tool collects the following information for each task:

- the time spent in each step of the process (study, shortlist, select and instantiate);
- the patterns that are browsed in the catalog;
- the patterns that are shortlisted and finally selected;
- the total time each pattern is looked at.

The tool is also equipped with a pause button, which the teams were instructed to use when they were taking a break.

Figure 1. Design of the experiment

Pressing this button stops the time measurements and at the same time hides all information that is visible in the tool, until work is resumed. The supervisors in the lab sessions reminded the teams of this functionality whenever necessary.

The collected measurements are stored in an encrypted log file, and are automatically transmitted to a web server after the completion of every task. The teams were made aware of this beforehand, so they knew they had to be connected to the Internet when completing a task. However, they did not know precisely which measures were collected.

To summarize, the design of the experiment is presented in Figure 1. Both the PLAIN and ANNOTATED groups have followed the same process (except for the augmentations described in Section IV-E), have used the same tool and performed the same set of tasks. The only difference between the groups were the features of the security pattern catalog. The ANNOTATED group teams had access to an annotated catalog, together with filtering functionality based on these annotations. The PLAIN group had access to the same catalog, but without annotations and filtering options. The design has been validated and calibrated in a small try-out before being used with the participants.

After the experiment, the teams have been asked to fill out an online questionnaire (see Section VI).

G. Measures and Experimental Hypotheses

First, we measure the *selection time* (T) as the time (in seconds) that is necessary for a team to arrive at their final pattern selection for each task. This measure includes the first three activities of the process, that is, studying the requirement, creating a shortlist of patterns and making the final selection. For each task, we test the (effective) null hypothesis that the mean selection time for both groups is equal:

$$H_0^T : \mu_T^A = \mu_T^P$$

where μ_T^A and μ_T^P are the average selection time of the ANNOTATED and PLAIN group, respectively.

Table III
NUMBER OF DATA POINTS FOR TIME T AND EFFICIENCY E

Task	Invalid	Outliers		Remaining			
		T	E	T		E	
				PLAIN	ANNO	PLAIN	ANNO
B	2	2	2	21	20	19	22
C	1	5	1	19	20	20	23
D	2	2	3	20	21	19	21
E	4	2	2	21	18	20	19
ALL	9	11	8	81	79	78	85

Second, we measure the *selection efficiency* (E) as follows. N_{view} is the total number of patterns that the team looked at (i.e., opened in the catalog browser) when performing a task, ignoring whether or not the pattern was retained or discarded for instantiation. N_{select} is the number of patterns that the team finally selected for instantiation in the architecture for that task. The selection efficiency then is $E = N_{select}/N_{view}$. As every selected pattern must have been viewed, $N_{view} \geq N_{select}$ and thus $0 \leq E \leq 1$. Note that $E = 1$ means that every viewed pattern was selected, maximizing efficiency. Again, for each task we test the (effective) null hypothesis that the mean efficiency for both groups is equal:

$$H_0^E : \mu_E^A = \mu_E^P$$

where μ_E^A and μ_E^P are the average selection efficiency of the ANNOTATED and PLAIN group, respectively.

V. DATA ANALYSIS

In the analysis, we ignore the warm-up task A, as it is heavily influenced by other factors such as the teams becoming familiar with the tool, the system and the process.

A. Data Validation and Outlier Removal

Before starting with the analysis of the data, we observed that in nine cases we did not obtain measurements for some tasks. Eight of these cases correspond to teams that decided that a task was already covered by the architecture. Hence, they skipped that task and no measurements about the execution of the rest of the process are available. Also, for one team, all data from task B was missing due to an occasional crash of the (otherwise very reliable) tool. These cases have been discarded.

Additionally, some of the obtained data points have been labeled as outliers, and excluded from further analysis. Data points are discarded when they are more than 1.5 times the inter-quartile range above (below) the upper (lower) quartile. The boxplots showing the outliers for both the time and the efficiency measures can be found in Figures 2 and 3, respectively. Concerning the time measure, across all tasks, 11 data points were labeled as outliers from the set of 171 available data points, resulting in a reduction of 6.4%. Concerning the efficiency measure, across all tasks, 8 data points were removed, resulting in a reduction of 4.7%. The

Figure 2. Boxplots of selection time T per task

Figure 3. Boxplots of efficiency E per task

number of remaining data points per task and treatment group is shown in Table III.

B. Summary of the Results

The descriptive statistics of the time T can be found in the left-hand side of Table IV. The results (see the ΔMean column) indicate that, on average, the ANNOTATED group has spent *more* time in most of the cases. This is a surprising result that goes against the expectations. Mind that tasks C–E are of medium to low difficulty. Task B is the most difficult one according to the experiment designers and, here, the difference between the two groups is negligible.

The descriptive statistics of the efficiency E can be found in the left-hand side of Table V. The results show that, on average, the ANNOTATED group is largely *more efficient* in

Table IV
ANALYSIS FOR THE SELECTION TIME T

Task	PLAIN		ANNOTATED		Δ Mean	p-value
	Mean (sec)	Stdev	Mean (sec)	Stdev		
B	1639	1141	1614	1170	−1.5%	.9897
C	1220	681	1648	804	+35.1%	.0817
D	1911	1072	2109	978	+10.4%	.4263
E	1373	859	1701	917	+23.9%	.1830

Table V
ANALYSIS FOR THE EFFICIENCY E

Task	PLAIN		ANNOTATED		Δ Mean	p-value
	Mean	Stdev	Mean	Stdev		
B	0.139	0.093	0.349	0.230	+151.1%	.00018
C	0.109	0.056	0.239	0.110	+119.3%	.00010
D	0.139	0.080	0.146	0.086	+5.0%	.7995
E	0.185	0.142	0.111	0.061	−40.0%	.1183

tasks B and C. The difference is much smaller in task D. These tasks are of medium to high difficulty. Task E is considered to be the easiest, as confirmed by the participants (see Section VI). In this case, the selection of the right pattern could have been done very efficiently by simply looking at the names of the patterns. Hence, using the annotations is likely to be an unnecessary complication. This could explain why the ANNOTATED teams underperformed in task E.

C. Statistical Analysis

In order to select a suitable statistical test, we use the Shapiro-Wilk normality test to determine whether the data follows a normal distribution. This is the case in task C for T and task D for E . In both cases, the two sample populations also have equal variance. When the data is normally distributed, we test the hypothesis by means of the (two-tailed) t-test, which is a parametric test. Otherwise, we use the (non-parametric) Wilcoxon rank-sum test (also known as Mann-Whitney U test). We always use $\alpha = 0.05$ as the significance level.

The resulting p -values of the statistical tests for time (T) and efficiency (E) are given in the right-hand side of Tables IV and V, respectively.

Concerning the time, none of the measured differences are statistically significant, and as such we cannot reject the null hypothesis H_0^T . Based on these results, we cannot conclude that the annotations had any effect on the mean selection time of the two groups.

Concerning the efficiency, the results are statistically significant for tasks B and C. This allows us to confidently reject the null hypothesis H_0^E and conclude that, for these two tasks, an improvement in efficiency was obtained by providing annotations in a pattern catalog.

Finally, we remark that we have also re-analyzed the data without removing the outliers, as discarding outliers is sometimes regarded as a controversial practice. However,

Table VI
NUMBER OF VIEWED AND SELECTED PATTERNS PER GROUP AND TASK

Task	Viewed patterns						Selected patterns					
	PLAIN			ANNOTATED			PLAIN			ANNOTATED		
	min	med	max	min	med	max	min	med	max	min	med	max
A	20	34.5	35	5	15	31	2	2	8	1	3	5
B	1	10	33	0	5	15	1	1	2	1	2	3
C	2	10	35	3	7	23	1	1	2	1	1	4
D	2	9	35	4	12	28	1	1	3	1	1	3
E	1	10	35	1	15	23	1	1	3	1	1	5

we report that we have obtained the same conclusions. Concerning T , the ANNOTATED group is slower in tasks B–E. However, the differences are not statistically significant. Concerning E , the same trends are observed: an advantage for the ANNOTATED group in tasks B and C, a tie in task D, and a disadvantage in task E. The differences in tasks B and C are still statistically significant.

D. Interpretation of the Results

The experimenters’ expectation, namely that the ANNOTATED group would be faster in selecting patterns because they can more easily focus on the relevant ones thanks to the guidance provided by the annotations, does not appear to hold. Nevertheless, for tasks B and C, there is a significant increase in efficiency, which shows that fewer irrelevant patterns are indeed browsed by the ANNOTATED group. To explain this effect, N_{view} and N_{select} (the contributors to the efficiency) should be investigated separately. Therefore, Table VI shows the number of patterns that was viewed and eventually selected by each group per task. The data show that for tasks B and C the ANNOTATED group viewed less patterns than the PLAIN group (N_{view} decreases), but retained more in their final selection (N_{select} increases). By applying the Wilcoxon test, these differences are found to be statistically significant for the selected patterns during task B (p -value 0.013) and C (0.015), and for the viewed patterns during task B (0.029). For the viewed patterns during task C, the decrease is close to being significant (p -value 0.051). The decrease in N_{view} indicates that the ANNOTATED group was more focused when choosing a suitable pattern, as they invested their time in gaining a deeper insight into a subset of the patterns. This could be the result of filtering the catalog based on the security objective annotations. Also, it is plausible that the relationship information led the ANNOTATED group to select additional (complementary) patterns, thereby increasing N_{select} .

E. Discussion

With time, it is natural that a team becomes more familiar with the solutions in the catalog. This effect is irrelevant in the previous discussion, due to the randomized order of the task execution. When the team uses the catalog for the first time, however, it is intuitively expected that the impact of the

annotations would be larger. To assess this effect, Table VI also contains the warm-up task A (first task for every team).

It is clear that the number of viewed patterns for task A is much higher for the PLAIN group than for the ANNOTATED group. Every team in the PLAIN group at least looked at 20 patterns when completing the first task, and half of the teams in that group looked at every pattern in the catalog. For the ANNOTATED group, the minimum and median are much lower, and no team looked at the entire catalog during task A. The difference between the two groups is statistically significant (Wilcoxon, $p = 1 \times 10^{-7}$). This demonstrates that annotations can have a beneficial impact when an architect is confronted with a pattern catalog for the first time, e.g., because she is dealing with a software quality she is less acquainted with. The annotations make it unnecessary to familiarize yourself with the whole catalog beforehand.

A legitimate question is how the correctness of a team’s solution was determined. The teams had to hand in a lab report for this part of the course, which was graded independently from the rest of the course itself. It is important to note that the grades have been assigned by six teaching assistants. As the graders were no stakeholders of the experiment or its outcome, they can be assumed to be unbiased. Also, the graders performed an up-front calibration of their grading criteria on two reports, in order to guarantee homogeneity in their scores. Reports have been graded on a scale from 0 to 20, which is familiar to the TAs. Overall, the average score obtained by all teams is 11.7 with the first quartile equal to 9.5. Hence, the bulk of the grades distribution is above the 50% threshold and it can be concluded that the teams did a fair job. The PLAIN group did slightly better than the ANNOTATED group by .7 points on average, but this difference is not statistically significant.

We have performed an analysis considering only the teams (about 30) that passed the 50% threshold in their grades. These are the teams that produced higher quality artifacts. We report that the same conclusions of Sections V-B and V-C can be drawn. The only difference concerns T : in task D the ANNOTATED group was faster (10%). Considering the 60% threshold (about 20 teams), the same trends are observed. However, the differences are more polarized.

Additionally, we performed an initial attempt at assessing the pattern selections made by the subjects, which is hard as there is no single correct baseline to compare against. We restrict ourselves to the most popular pattern selected by each group. The data show that, for each task, the most often selected pattern is the same in both groups. Hence, the annotations did not cause major differences with respect to the chosen pattern, but they might have influenced the completeness of the final solution. Further experiments are necessary in order to corroborate or refute this claim.

Table VII
QUESTIONNAIRE

Question	Answer
1. The tutorial and warm-up task were sufficient to become familiar with: (Strongly disagree/Disagree/Neutral/Agree/Strongly agree) Q1.1. The tool Q1.2. The architecture of the PMS system Q1.3. The process Q1.4. The security patterns catalog	Agree Agree Agree Agree
2. Rate your understanding of the task's description: (Very unclear/Unclear/Average/Clear/Very clear) Q2.(1–4). Task B–E	Clear
3. Were all the tasks of similar difficulty? (Yes/No)	No
4. Rate the difficulty of each task: (Very hard/Hard/Average/Easy/Very easy) Q4.1. Task B Q4.2. Task C Q4.3. Task D Q4.4. Task E	Average Average Average Easy
5. Indicate your agreement with these statements: (Strongly disagree/Disagree/Neutral/Agree/Strongly agree) Q5.1. The pattern-based process is intuitive Q5.2. The process puts too many constraints Q5.3. The catalog does not contain enough patterns Q5.4. The description of the patterns is clear Q5.5. The process wizard is user-friendly Q5.6. The UML editor is user-friendly Q5.7. The catalog browser is user-friendly Q5.8. Overall, the pattern-based process is useful	Agree Neutral Disagree Neutral Agree Disagree Disagree Agree
6. I am satisfied with the selected patterns. (Strongly disagree/Disagree/Neutral/Agree/Strongly agree) Q6.(1–4). Task B–E	Agree
7. How difficult was it to find a suitable pattern? (Very hard/Hard/Average/Easy/Very easy) Q7.(1–4). Task B–E	Easy

VI. QUESTIONNAIRE

At the end of the experiment, the teams were asked to fill in an online questionnaire. The purpose of the questionnaire is to validate the experimental assumptions and to gather feedback from the participants.

A. Validation of the Experimental Assumptions

A number of assumptions have been made while designing the experiment. For instance, we assume that the process is rather intuitive and hence natural to follow, or that the tool is easy to use. These assumptions guarantee that the measures are not altered by the experimental conditions. If not verified, the above assumptions are to be considered as threats to the validity of the results.

Tables VII and VIII present the questions asked to the participants, and the median of the answers (or the mode for yes/no questions). The full set of questions and answers is available online [7]. Note that the order of the questions have been rearranged in this paper for presentation purposes. A first assumption of the study is that the participants were adequately prepared when entering the experiment. The answers to Q1.(1–4) indicate that this was indeed the case. We also assume that the participants understood the tasks

they were assigned. This is very important for the sound construction of the experiment. The assumption is confirmed by the answer to Q2.(1–4).

Concerning the complexity of the individual tasks (questions Q3 and Q4), the participants report that tasks B–D are considered of equal difficulty. It looks like the higher difficulty anticipated by the experiment designers for task B (due to the fact that a system of patterns needs to be selected) is not particularly worrisome for the participants. Task E appears to be easier than the others (as expected). Tasks B and C lead to a statistically significant difference concerning the selection efficiency. Since they are of comparable nature according to the participants, the results obtained in the two tasks may reinforce each other.

We also assume that the process described in Section IV-E did not force the participants to adhere to a counterproductive workflow. From the answers to questions Q5.1, Q5.2 and Q5.8, we can conclude that the participants felt that the process was intuitive and useful.

A third assumption was that the security patterns catalog was sufficiently large to execute all the assigned tasks. This is confirmed by the answers to question Q5.3. However, the description of the patterns (which comes from the state-of-the-art) is reported to be less than ideal in question Q5.4. This confirms the concerns about the usability of the existing patterns, as mentioned in the introduction of this paper.

A further assumption tested in questions Q5.(5–7) is that the tool has no impact on the performance of the participants by being hard or counterintuitive to use. The answers to the questionnaire show that the tool was well absorbed by the participants, especially concerning the process wizard (Q5.5). They did complain about the limited usability of the Topcased UML editor (Q5.6), which is used to instantiate the selected patterns. According to the feedback, browsing UML diagrams was not an issue. Modifying the diagrams during step 4 was more tedious. Note, however, that this step is not used for the measurements. Hence, this issue has no effect on the outcome of the study. The participants of both groups were also slightly annoyed by one specific monitoring feature of the catalog browser. Pattern descriptions are organized into sections, e.g., problem, forces, solution, and so on. The tool records which section is accessed the most over each step (this data was not used in this paper). To this aim the sections must be opened via a mouse click and they close automatically after 60 seconds.

B. Additional Feedback

Per task, we asked the teams to rate their level of satisfaction with respect to the solution they selected (Q6). We also asked about the level of perceived difficulty related to the selection (Q7). The teams agree that they found a satisfactory pattern for each task (median is 4 in a satisfaction scale of 5). They also considered the selection process as easy (median is 2 in a difficulty scale of 5).

Table VIII
QUESTIONNAIRE CONT'D (ANNOTATED GROUP ONLY)

Question	Answer
8. To what degree was the following information helpful to <i>shortlist</i> a pattern in step 2? (Not helpful at all/Not helpful/Neutral/Helpful/Very helpful)	
Q8.1. Trade-off labels	Neutral
Q8.2. Relationships among patterns	Helpful
Q8.3. Security objectives	Very helpful
Q8.4. Applicability	Neutral
9. To what degree was the following information helpful to <i>select</i> a pattern in step 3? (Not helpful at all/Not helpful/Neutral/Helpful/Very helpful)	
Q9.1. Trade-off labels	Helpful
Q9.2. Relationships among patterns	Helpful
Q9.3. Security objectives	Helpful
Q9.4. Applicability	Neutral

For the ANNOTATED group only, we also gathered some feedback about the perceived usefulness of the annotations (questions 8 and 9). Overall, the participants see less value in the ‘applicability’ annotation. On the contrary, the connection between requirements and solutions via the ‘security objectives’ is very much appreciated, especially in step 2.

VII. THREATS TO VALIDITY

A. Internal Validity

The familiarity of the participants with the concepts used in the experiment (patterns, misuse cases, quality scenarios, UML, and so on) might have an impact on the participants’ performance. We tried to mitigate this threat by providing extensive training on all the matters during the class hours. However, we cannot guarantee that all participants attended all the necessary classes.

Also, the ANNOTATED group has received about half an hour of additional lecture to make them familiar with the annotations. However, as the lecture only explained the nature of the annotations and how to use them, and not the pattern catalog nor the case study, we believe that the impact on the results is negligible.

The participants were not supervised at all times, as they were allowed to work at home. Further, the experiment spanned over two labs sessions. These are opportunities for treatment diffusion, as the teams might have interacted.

Furthermore, the number (and type) of patterns in the catalog might have influenced the results. E.g., organization and navigation might become more helpful for larger catalogs, resulting in a bigger difference between the groups.

B. External Validity

The main issue threatening the generalization of the results concerns the use of master students instead of professionals. However, it is generally advised to test new theories starting with students via exploratory studies [20], [21]. Further, Runeson observes that differences can be small between graduate students (our case) and professionals [22].

The results might also be specific for the selected set of annotations. Although in Section III we have shown that the set is representative of the related work, replica studies are likely to be needed in order to establish whether alternative annotations would lead to similar conclusions.

The tasks are of a small size with respect to real-world design endeavors. Larger tasks also require longer time for the experiment execution, to the further detriment of the control on the experiment itself. Given the time-frame of the experiment (2 lab sessions), the used tasks represent the best compromise. However, we acknowledge that the study should be replicated with larger tasks.

The results might be influenced by the objects used in this study. For instance, the PMS system is a realistic application, but one of moderate size. Also, the experiment was performed only using this architecture. This represents a drawback in terms of the realism of the experiment. On the other hand, some interfering issues (such as getting to know the system) are reduced thanks to the limited size.

Finally, the tool that was used to browse the patterns had a few counterintuitive constraints, to make it possible to monitor the subjects’ actions. As reported in the questionnaire, this has hindered the subjects, with possibly a negative impact on their performance or motivation.

VIII. CONCLUSIONS

We have performed a controlled experiment with 90 participants to determine the impact of annotations for security patterns on the performance of a software architect. Surprisingly, the results do not show an advantage in terms of the time that is necessary to complete a design task. On the other hand, the use of annotations noticeably reduced the number of irrelevant patterns that were considered in two out of the four cases we investigated. This result is statistically significant and applies to a master student population and for tasks of average difficulty.

The study presented in this paper did not single out one specific annotation. Rather, a representative set has been studied jointly. The main lesson learned is that it is not self-evident that annotations provide an edge. Therefore, we recommend that new proposals are carefully evaluated.

Starting from the additional data collected in this experiment but not used in this paper, we are also in the process of analyzing the quality of the design artifacts produced by the teams. We expect to gather useful insight on the effect of the annotations on the architectural choices as well as the overall soundness of the adopted security solutions.

ACKNOWLEDGEMENT

This research is partially funded by the EU FP7 project NESSoS, the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund KU Leuven.

REFERENCES

- [1] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering*. Wiley, 2006.
- [2] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, "An analysis of the security patterns landscape," in *International Workshop on Software Engineering for Secure Systems (SESS)*, 2007.
- [3] M.-A. Laverdiere, A. Mourad, A. Hanna, and M. Debbabi, "Security design patterns: Survey and evaluation," in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2006.
- [4] M. Hafiz and R. Johnson, "Security patterns and their classification schemes," Microsoft's Patterns and Practices Group, Tech. Rep., 2006.
- [5] M. Hafiz, P. Adamczyk, and R. Johnson, "Organizing security patterns," *Software, IEEE*, vol. 24, no. 4, pp. 52–60, 2007.
- [6] R. Scandariato, K. Yskout, T. Heyman, and W. Joosen, "Architecting software with security patterns," K.U.Leuven, Tech. Rep. CW515, April 2008.
- [7] K. Yskout, R. Scandariato, and W. Joosen, <http://distrinet.cs.kuleuven.be/software/securitypatterns>.
- [8] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, vol. 5, no. 5, pp. 35–47, 2008.
- [9] C. Steel and R. Nagappan, *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management*. Prentice Hall, 2006.
- [10] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 26, no. 3, 1987.
- [11] M. Howard and S. Lipner, *The Security Development Lifecycle*. Microsoft Press, 2006.
- [12] D. G. Rosado, C. Gutierrez, E. Fernandez-Medina, and M. Piattini, "Security patterns related to security requirements," in *International Workshop on Security in Information Systems (WOSIS)*, 2006.
- [13] O. Mazhelis and A. Naumenko, "The place and role of security patterns in software development process," in *International Workshop on Security in Information Systems (WOSIS)*, 2006.
- [14] M. VanHilst, E. B. Fernandez, and F. Braz, "A multi-dimensional classification for users of security patterns," *Journal of Research and Practice in Information Technology*, vol. 41, no. 2, 2009.
- [15] D. Hatebur, M. Heisel, and H. Schmidt, "Analysis and component-based realization of security requirements," in *International Conference on Availability, Reliability and Security (AREs)*, 2008.
- [16] E. Fernandez, H. Washizaki, N. Yoshioka, A. Kubo, and Y. Fukazawa, "Classifying security patterns," in *Asia-Pacific web conference on Progress in WWW research and development (APWeb)*, 2008.
- [17] M. Weiss, "Modeling security patterns using NFR analysis," in *Integrating security and software engineering: advances and future visions*, H. Mouratidis and P. Giorgini, Eds. IGI Global, 2007.
- [18] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley, 2003.
- [19] A. O. Guttorm Sindre, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, January 2005.
- [20] W. Tichy, "Hints for reviewing empirical work in software engineering," *Empirical Software Engineering*, vol. 5, no. 4, December 2000.
- [21] J. Carver, L. Jaccheri, and S. Morasca, "A checklist for integrating student empirical studies with research and teaching goals," *Empirical Software Engineering*, vol. 15, no. 1, February 2010.
- [22] P. Runeson, "Using students as experiment subjects - an analysis on graduate and freshmen student data," in *International Conference on Empirical Assessment in Software Engineering (EASE)*, 2003.