Information and Software Technology 54 (2012) 1029–1043

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Evaluation of the Pattern-based method for Secure Development (PbSD): A controlled experiment

Jenny Abramov [a,b], Arnon Sturm [a,*], Peretz Shoval [a]

[a] Dept. of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[b] Deutsche Telekom Laboratories, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

ARTICLE INFO

Article history:
Received 26 April 2011
Received in revised form 7 February 2012
Accepted 20 April 2012
Available online 28 April 2012

Keywords:
Database design
Model driven development
Secure software development
Authorization
Security patterns
Controlled experiment

ABSTRACT

Context: Security in general, and database protection from unauthorized access in particular, are crucial for organizations. Although it has been long accepted that the important system requirements should be considered from the early stages of the development process, non-functional requirements such as security tend to get neglected or dealt with only at later stages of the development process.
Objective: We present an empirical study conducted to evaluate a Pattern-based method for Secure Development – PbSD – that aims to help developers, in particular database designers, to design database schemata that comply with the organizational security policies regarding authorization, from the early stages of development. The method provides a complete framework to guide, enforce and verify the correct implementation of security policies within a system design, and eventually generate a database schema from that design.
Method: The PbSD method was evaluated in comparison with a popular existing method that directly specifies the security requirements in SQL and Oracle's VPD. The two methods were compared with respect to the quality of the created access control specifications, the time it takes to complete the specification, and the perceived quality of the methods.
Results: We found that the quality of the access control specifications using the PbSD method for secure development were better with respect to privileges granted in the table, column and row granularity levels. Moreover, subjects who used the PbSD method completed the specification task in less time compared to subjects who used SQL. Finally, the subjects perceived the PbSD method clearer and more easy to use.
Conclusion: The pattern-based method for secure development can enhance the quality of security specification of databases, and decrease the software development time and cost. The results of the experiment may also indicate that the use of patterns in general has similar benefits; yet this requires further examinations.

1. Introduction

Data is the most valuable asset for an organization as its survival depends on the correct management, security, and confidentiality of the data [11]. In order to protect the data, organizations must secure data processing, transmission, and storage. Nowadays, organizational systems are developed with minor emphasis on security aspects; system developers tend to neglect security requirements or deal with them only at the end of the development process. In [21] the authors stated that "*software engineers and practitioners should assimilate basic security techniques and discover new techniques for integrating them in the current practice, while understanding associated costs and benefits*."

Various efforts to deal with the security aspect at the early stages of the software development life cycle have been made. These include various extensions to UML that enable modeling of security features such as UMLsec [16], threat and attack modeling techniques such as misuse cases [31], methods for requirements engineering such as Secure Tropos [27], and methodologies for developing secure applications and databases such as [12,13]. Nevertheless, such methods primarily provide guidelines of how to deal with security within a certain stage of the software development process, or address only specific security-related aspects or mechanisms in the development process. Moreover, existing methods do not deal with organizational security policies and guidelines. To the best of our knowledge, no existing method proposes a complete framework that provides guidelines regarding organizational security policies, imposes them on a system design, and transforms the design to code.

* Corresponding author.
E-mail address: sturm@bgu.ac.il (A. Sturm).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.04.001

To overcome existing limitations, we have developed a Pattern-based method for Secure Development – PbSD. In this paper we focus on the development of secure databases. The method deals with both the organizational level and the application development level. At the organizational level, organizational security policies are defined by security experts in the form of security patterns. Security patterns emerge from the classic idea of *design patterns* introduced by Gamma et al. [14]. Design patterns capture expert knowledge about a recurring problem and provide a well-proven solution to it. Since patterns are independent of specific applications, they can be used to assist in the development of numerous applications. At the application development level, the designers create (among other things) a conceptual data model. According to the PbSD method, this conceptual data model is refined to include the security requirements using the security patterns; later on the conceptual data model is verified vis-à-vis the organizational security policies (the patterns); finally, a secure database schema is automatically generated.

In this paper we examine whether the use of PbSD results in more secured database specifications. To do so, we compared the method in a controlled experiment versus regular usage of specifying access control privileges on database schemata. Our hypotheses are that the use of the PbSD method provides better quality access control specifications, reduces efforts to specify these specifications, and increases developers' satisfaction.

The rest of this paper is structured as follows: Section 2 provides a review of related studies. Section 3 introduces the pattern-based method for secure development. Section 4 elaborates on the evaluation of the method. Finally, Section 5 summarizes and set plans for future research.

2. Related studies

In this section we first review studies on the evaluation of design patterns. Then, we review existing approaches for secure software development, with emphasis on Model Driven Development (MDD) [42].

2.1. Evaluation of design patterns

As our approach is based on patterns, we first review studies on the evaluation of design patterns. Prechelt et al. [35] conducted a controlled experiment to assess whether designs using patterns improve maintenance even if the actual design problem is simpler than that solved by the design patterns. Although the findings were dependent on the situation, the authors recommend applying design patterns as they provide flexibility (unless there is a clear reason to prefer the simpler solution) since unexpected new requirements often appear. The experiment was replicated by Vokáč et al. [45], in which a programming environment was used. The results indicate that some patterns are easier to understand and use than others. This study led to the conclusion that design patterns cannot be characterize as useful or harmful in the general sense with respect to maintenance.

Ampatzoglou and Chatzigeorgiou [3] evaluated the use of object-oriented design patterns in the domain of game development. They examined two open-source games for two versions of the game, one with design patterns under study and the other without. The authors concluded that patterns can be beneficial with respect to maintainability (as complexity and coupling were reduced), and that patterns tend to increase cohesion.

Zhang and Buden [48] conducted a comprehensive literature review in the form of a mapping study on the effectiveness of design pattern as a mechanism for transferring knowledge. The authors found support for the claim that patterns have a positive impact upon maintenance, that there is no support for the claim that patterns help novices to learn about design, and that there is a limited support for increased productivity when using patterns.

Reinhartz-Berger and Sturm [37] examine a domain model (which is similar to a pattern) impact on the creation of application artifacts. They found that the availability of domain artifacts (i.e., models) helps achieve more complete application artifacts (i.e., models).

To summarize, it seems that only limited efforts were done in order to evaluate the usage of patterns. This is also confirmed by Chen and Babar [6] who surveyed studies that deal with software product line engineering methods, which play similar role as patterns. The authors found that only a few studies employee empirical evaluation to analyze the effectiveness of these methods.

Our study, presented in this paper, can also be considered as a contribution in the area, in the sense that it examines the effectiveness of a pattern based design.

2.2. Review and evaluation of secure software development methods

Various techniques for secure software development exist; they vary from threat modeling (problem space) to design (solution space). Usually, these techniques adopt principles from common software development approaches and extend them to the security domain. In the following we review and evaluate some representative studies on such techniques.

Attack trees and misuse cases are techniques for elicitation and analysis of security threats. Opdahl and Sindre [31] conducted a pair of controlled experiments that compared the two techniques. The subjects were asked to solve two threat identification tasks using the two techniques. The experiment evaluated the *effectiveness* of the techniques by the number of threats found, the *coverage* of the techniques by the found threats types, and the *perceived qualities* of the techniques by a post-task questionnaire. The results showed that attack trees were more effective for finding threats, especially for authorization and confidentiality threats. Yet, both techniques obtained equivalent scores regarding participants' opinions on the technique.

Secure Tropos [27] is a security oriented extension to the goal-driven requirements engineering methodology Tropos. Secure Tropos organizational model is based on i^* [47], adopting its basic concepts such as actors, goals, tasks and social dependencies. It has been evaluated through several case studies. In [24] the authors analyzed a banking application while using Secure Tropos for detecting vulnerabilities caused by conflicts among high-level functional and security requirements. Another case study [7] presented support for capturing security and privacy requirements in the health care domain; in that case study, a pattern library developed within the SERENITY project [43] was used. First, the authors showed how to define security patterns at the organizational level; then they showed how these patterns are used to support the analysis of organizational security and privacy requirement.

UMLsec [16] extends UML to enable specifying security concerns in the functional model. It uses standard UML extension mechanisms, stereotypes with tagged values to formulate the security requirements, and constraints to check whether the security requirements hold in the presence of particular types of attacks. It is supported by a tool for automated verification at the design level [15]. Several case studies were performed to evaluate UMLsec. An application of UMLsec in the health care domain is presented in [17], where a smart-card based architecture was analyzed with the use of UMLsec tools. Another case study [22] presented the application of UMLsec to a biometric authentication system. UMLsec was found to be effective in specifying security requirements.