

S&D Pattern Deployment at Organizational Level: A Prototype for Remote Healthcare System

Pierre Busnel⁵

University of Sherbrooke, Canada

Paul El Khoury¹

SAP Research & LIRIS, University of Lyon I, France

Keqin Li³

SAP Research, France

Ayda Saidane⁴

University of Trento, Italy

Nicola Zannone²

University of Toronto, Canada

Abstract

The analysis of security incidents and frauds has shown that several vulnerabilities of IT systems are due to loopholes in the policies and procedures adopted by organizations as well as in their structure. Organizations have thus to address security and dependability issues by analyzing their organizational setting. In this paper, we present a methodology to support the deployment of Security & Dependability patterns according to their position in the Enterprise Architecture and the underlying system infrastructures. The methodology discriminates the pattern deployment process between recommendations and guidelines. Recommendations concretize the deployment with refined software and/or hardware related patterns, whereas guidelines specify the organizational patterns in terms of the system-to-be, proposing human-resource and/or policy solutions. To make the discussion more concrete, we illustrate the framework with a case study on an emergency scenario within a remote healthcare system.

Keywords: Security & Dependability Patterns, Organization, Pattern Deployment, Healthcare System.

¹ Email: paul.el.khoury@sap.com

² Email: zannone@cs.toronto.edu

³ Email: keqin.li@sap.com

⁴ Email: ayda.saidane@unitn.it

⁵ Email: pierre.busnel@usherbrooke.ca

1 Introduction

The analysis of security incidents and frauds has revealed that security is often compromised by exploiting organizational vulnerabilities [4,9,7]. Business organizations and government agencies need to adopt appropriate security measures including security policies and procedures intended to protect their IT systems as well as to support their management and use.

The last years have seen a growing effort to integrate security into the system development process through authentication and access control mechanisms [10,2]. However, they focus on the IT system solely, and do not address security issues within the organizational setting. Unfortunately, such an approach leaves a gap between security measures and the requirements of the organization where the IT system is embedded.

It is generally accepted in the Requirements Engineering community that system development requires to analyze the system-to-be together with its intended operational environment [14]. This is even more important when the system has to meet security and dependability requirements since security breaches often occur at the organizational level rather than at the technical level [1]. At the organizational level, the system is seen as a set of interacting agents (organizations/humans/software agent); each of them is in charge of achieving a set of business goals. At this level of abstraction, the focus is on what is needed to achieve organizational goals such as task and resource allocation with the necessary authorizations, delegations or trust relations. Functional and security solutions related to the processes used to achieve these goals fall out of the scope of our study on the deployment of organizational security patterns. These studies are addressed at business and communication levels using different types of formalism and technology (e.g., Workflow, Web Service, Communication Channels, etc.).

Security & Dependability (S&D) patterns provide means to capture security expertise by suggesting solutions to recurring security and dependability issues [16]. Deploying an S&D pattern at the organizational level corresponds to modify an initial organizational structure where S&D requirements are not fulfilled, leading to a new organizational structure where these requirements are fulfilled. This makes it possible to reduce vulnerabilities in the policies adopted by an organization and facilitate the selection of appropriate technical solutions during the subsequent phases of the development process.

Current methodologies for applying security solutions at the organizational level are limited to the conceptual level. In this direction, we find the proposal by Kim et al. [11], who introduces methods to reduce threats to the system by applying security engineering. The proposed method for building security countermeasures does not comprise details on its application into the systems to be enhanced. An interesting work on patterns was provided by Konrad et al. [12]. The aim of this work is to implement efficient security and privacy solutions as an integral part of a business process and to deliver value to customers by measuring security risks. Yet, these patterns are only available at the workflow (as opposed to organizational) level and are not validated. In [16] the author takes advantages of the pattern approach to propose a set of security solutions to be applied during the development process.

Similarly, the proposal by Yoder and Barcalow [17] covers different security solutions, such as role-based access control, but lacks technical information sufficient to assist system developers in their implementation. However, this conceptual approach (i.e., starting from formal specifications for ending up with executable code) is not suitable for securing an organization since it focuses more on code generation leaving little room for the analysis of the context where solutions are applied.

In this paper, we present a flexible and generic approach for the deployment of S&D patterns at the organizational level. This flexibility results from the adoption of three complementary representations of S&D patterns. The first patterns representation is conceptual and is used to (proof-of-concept) validate S&D solutions. The second representation concerns the infrastructure and describes the context in which solutions are applied. Finally, patterns are represented through the guidelines driving their implementation. We, thus, use the term “S&D pattern” through the paper to refer to any of above three representations. For pattern definition and validation, we have taken advantages of the S&D pattern library defined in the SERENITY project⁶ [3].

In this study, we bound the nature of systems to Software, Hardware, Human Resource or Policy infrastructure and propose deployment guidelines determined upon these types of infrastructure. These guidelines describe how patterns should be implemented. The aim of the proposed approach is to enhance the best practices for the definition of security policy and procedure as well as for the selection of the infrastructure that support them. We demonstrate the effectiveness and feasibility of our approach using a case study in which S&D patterns are deployed in a remote healthcare system prototype.

The paper is structured as follows: Section 2 introduces a healthcare system that is used as a running example throughout the paper. Section 3 presents an overview on the background underlying our methodology and illustrate them with examples driven from our case study. Section 4 describes the methodology proposed in this paper, illustrated with the remote healthcare system. Finally, Section 5 concludes the paper with some directions for future work.

2 Case Study: remote healthcare system

The development of healthcare systems raises a number of S&D issues. The objective of this typology of systems is to monitor the patient health status and provide the necessary assistance. To this end, healthcare systems should support the interaction and collaboration between doctors, pharmacists, patients, social workers and emergency medical teams especially during emergency situations.

Patient health condition can be monitored through various wearable medical sensors worn as a washable smart T-shirts [8]. All these sensors form the Body Sensor Network (BSN). The measured data are collected and pre-processed by a personal mobile hub such as a Personal Digital Assistant (PDA). Similarly, the patient house is equipped with a sensor network and a local server, which centrally processes the sensor data, for monitoring the activity of the patient and the envi-

⁶ EU-IST-IP 6th Framework Programme - SERENITY 27587 - <http://www.serenity-project.org>.

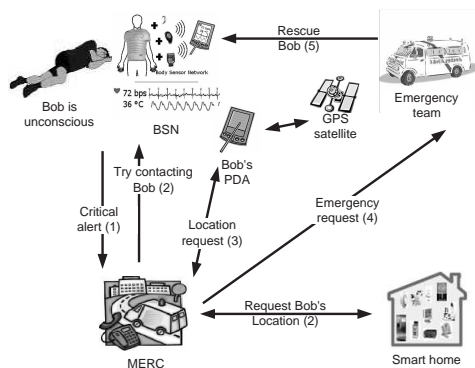


Fig. 1. Emergency case study

ronmental setting. Hereafter, we refer to it as the smart home. A concrete example of the smart home is the house built by the Domus Laboratory at the University of Sherbrooke [15]. The information collected by the BSN and smart home are sent to the Monitoring and Emergency Response Centre (MERC), the organization responsible for the maintenance and storage of patient medical data, such as the Electronic Health Record (EHR). The MERC processes such data to have a constant snapshot of the patient health status and promptly initiate proper healthcare procedures when a potential emergency alert is identified.

Among the possible application scenarios in the remote healthcare system, we focus here on an emergency situation. In such a scenario, the patient health status and activities are constantly monitored by the BSN and the smart home. All alerts and critical situations detected are sent to the MERC where, after a detailed analysis, the corresponding actions to help the patient are triggered. Hereafter the scenario is described in detail.

A patient, Bob, is equipped with a BSN. Bob's BSN detects a critical drop of blood pressure. This information is transmitted by the BSN to the patient's PDA.⁷ The PDA immediately interprets this as an emergency situation and sends an alert to the MERC as depicted in Figure 1. Alerts may also be voluntarily sent by Bob to request doctor assistance, for instance, when he feels giddy. In the case of alert, a rescue request together with Bob's location information are sent by the MERC to the emergency team who acknowledges to it. The emergency team is given access to Bob's EHR and last medical data collected by the BSN. When Bob is found, the emergency team sends a notification to the MERC together with comments regarding medicines administrated to Bob before arrival.

The analysis of the emergency scenario have raised many security and dependability issues concerning the safety of the patient and the protection of his data. Table 1 lists the main requirements extracted from this scenario. In this work, we mainly focus on S&D requirements to be satisfied at the organizational level.

⁷ A PDA runs an eHealth software designed to support medical request and report sending and receiving performed by the actor and to be compliant with the MERC.

Req 1	If the patient loses consciousness, his e-health terminal shall receive enough data from the patient smart T-shirt to immediately figure out the dangerous status and alert the MERC.
Req 2	If the emergency procedure has started, the MERC shall discover a medical team or a doctor that commits to rescue or assist the patient.
Req 3	The MERC's reception of the alert messages sent by the Smart Home and Bob's eHealth terminal shall be reliable.
Req 4	The system shall guarantee the non-repudiation on commitment of the medical team and doctor
Req 5	Each communication between the MERC and the eHealth terminals of the selected doctor and the medical team shall guarantee integrity and confidentiality of the data exchanged.
Req 6	Each communication between the eHealth terminals of the selected doctor or the medical team with the patient eHealth terminal shall guarantee integrity and confidentiality of the data exchanged.
Req 7	Insertions, modifications made by the selected doctor and medical team to the patient's EHR shall be reliable.

Table 1
Sample of S&D requirements for emergency scenario

3 Preliminaries

3.1 S&D patterns

Patterns have been proposed in the Software Engineering mainstream as a method for object-based reuse [5]. S&D patterns are best practices capturing knowledge of S&D experts and making it available to system developers. This transfer of knowledge is intended to improve the quality of the developed systems from S&D point of view. Along the line suggested in our previous work [3], we define S&D patterns as a quadruple $\langle Context, Requirement, Solution, Consequences \rangle$ where

Context defines the situation and conditions of pattern applicability. These conditions are related to the minimal set of agents and their relationships necessary to highlight the problem (threats) causing the non-fulfillment of the S&D requirements. They should be minimized in order to make the pattern more generic.

Requirement is the expression of the needs of a system to enhance its dependability or security level. The description of the requested S&D properties may also be expressed in terms of the acceptable frequency and severity of failure modes.

Solution specifies how the requirements are achieved. It defines the needed modifications (addition/elimination of agents/resources) to be applied to the organizational structure of the system in order to implement the appropriate S&D mechanism that ensures the fulfillment of the requirements.

Consequences highlights the effects of the pattern implementation on the system behavior. It is important to make aware system developers about possible side effects caused by the pattern application. For example, introduction of vulnerabilities, high costs, legal status (i.e., if compliant to some legislation or not), bad impact on other S&D properties or information about performances.

Patterns are defined in general terms as they should be applicable in different application domains. For instance, in the SERENITY project they have been applied to an air traffic management system, a loan process system, and an e-Government portal besides the remote healthcare system presented in this paper. The organizational S&D patterns in the SERENITY pattern library are described using SI* [13], a modeling language for designing secure socio-technical systems. In particular, SI*































VA Enterprise Architecture	DATA What	FUNCTION How	NETWORK Where	PEOPLE Who	TIME When	MOTIVATION Why	Based on work by John A. Zachman
SCOPE (CONTEXTUAL) <i>Planner</i>	Things Important to the Business  Ent = Class of Business Thing Rel = Business Relationship	Process Performed  Function = Class of Business Process	Business Locations  Node = Major Business Locations	Important Organizations  People = Major Organizations	Events Significant to the Business  Time = Major Business Event	Business Goals and Strategy  Ends/Means = Major Business Goals	SCOPE (CONTEXTUAL) <i>Planner</i>
ENTERPRISE MODEL (CONCEPTUAL) <i>Owner</i>	Semantic Model  Ent = Business Entity Rel = Business Relationship	Business Process Model  Proc = Business Process I/O = Business Resources	Business Logistics System  Node = Business Location Link = Business Linkage	Work Flow Model  People = Organization Unit Work = Work Product	Master Schedule  Time = Business Event Cycle = Business Cycle	Business Plan  End = Business Objective Means = Business Strategy	ENTERPRISE MODEL (CONCEPTUAL) <i>Owner</i>
SYSTEM MODEL (LOGICAL) <i>Designer</i>	Logical Data Model  Ent = Data Entity Rel = Data Relationship	Application Architecture  Proc = Application Function I/O = User Views	Distributed System Architecture  Node = System Component Link = System Linkage	Human Interface Architecture  People = User Work = User Action	Processing Structure  Time = System Event Cycle = Processing Cycle	Business Rule Model  End = Structural Assertion Means = Action Assertion	SYSTEM MODEL (LOGICAL) <i>Designer</i>
TECHNOLOGY MODEL (PHYSICAL) <i>Builder</i>	Physical Data Model  Ent = Segment/Tuple Rel = Primary Key	System Design  Proc = Computer Function I/O = Data Elements/Sets	Hardware Architecture  Node = Hardware Component Link = Hardware Linkage	Software Architecture  People = Software Component Work = Software Product	Architecture  Time = Software Event Cycle = Component Cycle	Rule Design  End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) <i>Builder</i>
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) <i>Sub-Contractor</i>	Data Definition  Ent = Field Rel = Address	Program  Proc = Language Statement I/O = Control Block	Addressing  Node = Addresses Link = Protocols	Identity Work  People = Identity Work = Job	Interrupt  Time = Interrupt Cycle = Machine Cycle	Rule Design  End = Sub-Condition Means = Step	DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) <i>Sub-Contractor</i>

Fig. 2. Zachman Framework for Enterprise Architecture

has been used to graphically represent the context and solution of patterns. These patterns have been (proof-of-concept) validated through an automated reasoning technique [6] underlying SI*. Such a technique is used to verify if patterns satisfy the requirements for which they have been designed. A detailed description of SI* and of the underlying formal analysis techniques, however, goes clearly beyond the scope of this paper and we refer to [6,13] for them.

Example 3.1 (Redundancy Pattern) *The first and basic event in the emergency scenario is the alert triggered by the smart home and PDA to the MERC. As specified in Req 3 of Table 1, the reception of triggering alerts from both the smart home and eHealth terminal shall be reliable. The system designer may select the Redundancy pattern from the SERENITY library to increase the level of system reliability. The proposed solution consists in duplicating unreliable services.*

3.2 Framework for Enterprise Architecture

SERENITY patterns are generic enough to be applicable into several application domain. When used in particular cases, they need to be customized and adapted taking into account functional requirements and constraints specific to the application domain. Aligned with the remote healthcare system presented in Section 2, our objective is the deployment of organizational S&D patterns to support enterprises in their daily business activities.

John Zachman has developed “the Zachman Framework for Enterprise Architecture” [18] that, similarly to the periodic table in chemistry, defines a roadmap for IT and business architectures using the information collected about companies. According to [18], “Enterprise” is an identified set of interacting business functions capable to operate as an independent, standalone entity. Therefore, enterprises can be recursively defined. Architecture provides the underlying framework technology while remaining implementation independent.

The Zachman framework organizes processes with respect to the perspective of

various players. These players, comprising the rows in the matrix of the framework, include: (1) *Planner*, who has undertaken to do business in a particular industry. This row defines the boundaries of the company as included, relevant, necessary or excluded; (2) *Owner*, who represents the business people that run the organization; (3) *Designer*, who creates the concepts quite independently from technologies to solve the problems of the business; (4) *Builder*, who defines the company implementation based on the previous row's constraints; and finally (5) *Sub-Contractor*, who specifies the implementations to specific technology products being used for implementation. Each of these rows corresponds to a user visualization of the Data, Function, Network, People, Time, and Motivation.

Example 3.2 (System Design) *Following the lines of Example 3.1, we study Req 3 in Table 1 from the perspective of the Builder (row 4). The Builder of the system defines the company implementation based on the concepts proposed to solve the problems of the business. Req 3 specifies the reliability on the reception of the alerts triggered by a PDA and a smart home, by the MERC. This requirements falls in the functional column of the matrix in Figure 2 as some technological choices has to be taken. Therefore, the representation of the MERC's reliability is through the unique cell made by row 4 and column 2.*

3.3 Organizational Infrastructure

An organizational infrastructure is the implementation platform used by an organization to perform its business activities. Here we mainly focus on four types of infrastructures: Software, Hardware, Human Resource, and Policy infrastructures.

- (i) *Software infrastructure* comprises the IT system within the organization. It targets the data maintenance and data processing performed by organizations. Examples of software infrastructures are Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Service Oriented Architecture (SOA).
- (ii) *Hardware infrastructure* covers the infrastructure usually made by sensors and devices. It describes the new trend towards pervasive computing, where Ambient Intelligent (AmI) ecosystem provides facilities to better support organizations in their activities.
- (iii) *Human Resource infrastructure* comprises the humans within the organization and describes their roles and interactions with the other parts of the system. Organizations are usually technique-mediated. Humans utilize technical media such as emails, telephones, management IT systems, video cameras and other software and hardware based tools, to run organization businesses. They are often the users of the IT system, and the participation of system users usually defines the success rate for these businesses.
- (iv) *Policy infrastructure* comprises organizational policies as well as laws and regulations which the organization must comply with. It consists of authority regulatory documents, contracts, certificates, etc.

These four types of infrastructure play a key role in the deployment process of the organizational S&D patterns.

Example 3.3 (Monitoring and Emergency Response Centre) *The MERC is a clear example of hybrid infrastructure as it involves humans (e.g., administrative staff, doctors, etc.), hardware (e.g., PDAs), and software applications. For the sake of simplicity, we mainly focus on the infrastructures necessary to satisfy Req 3 and Req 7. Assuming that Bob has given his consent to the MERC to collect and process his medical data, the MERC (in the position of ‘Sub-Contractor’ at row 5 with respect to the Zachman framework in Figure 2) handles the data insertion and modification in the emergency scenario processes through a software infrastructure. This software infrastructure reflects the IT implementation used between Bob’s eHealth terminal and the MERC’s workflow engine. Namely, the EHR’s implementation at the MERC should be provided in SOA for web services. Hence, we realize that Req 7 specifies that operations made within this infrastructure shall be reliable. Req 3 tackles the reliability of alerts sent to the MERC; while the smart home and eHealth terminal will mainly be hardware and software infrastructures their receiver from MERC’s side is judged according to the ‘Builder’ at row 4 to be human based. Such a decision is aligned with legislation where big interest in putting responsibility or having sufficient explanations when medical problems occur.*

4 Methodology

In this section, we propose a methodology to deploy organizational S&D patterns (Figure 3). This methodology is based on the library of S&D organizational patterns defined in the course of the SERENITY project (see [3]). The solution provided by those patterns describes a new organizational structure that satisfy some requirements demanded by the system. Basically, S&D patterns present general solutions in terms of new goals to be achieved by some actor and social relations among actors within the system. Patterns are then selected from the S&D pattern library according to the requirements to be satisfied by the system and its environmental context. However, solutions do not make any reference to implementation infrastructures. Not surprisingly, this abstraction level allows the application of additional (combination of) solutions available at a lower operational level (e.g., web service and workflow). Even with this additional constraint, we believe that the interpretation of the pattern solution can be satisfied by the expert interpretations. For instance, there may exist several possible implementations of the same pattern.

The right part of Figure 3 describes the methodology to deploy organizational S&D patterns. We take into account three additional parameters in order to create and/or retrieve⁸ deployment procedures: `MODELER PROFILE`, which designates the role of the pattern requester, taken from the set of roles presented by the Zachman framework; `MODEL POSITION IN ENTERPRISE ARCHITECTURE`, which indicates the purpose of its usage, driven by the purposes presented in the Zachman framework; and `MODEL’S IMPLEMENTATION INFRASTRUCTURE` which points out the infrastructure (out of the four infrastructures we identified in Section 3.3) where the pattern will be used in the system-to-be. These parameters enrich the pattern solution with information related to their deployment in the enterprise architecture.

⁸ Security experts follow this methodology to specify deployment procedures of their patterns, while requesters also do so for retrieving these deployment procedures.

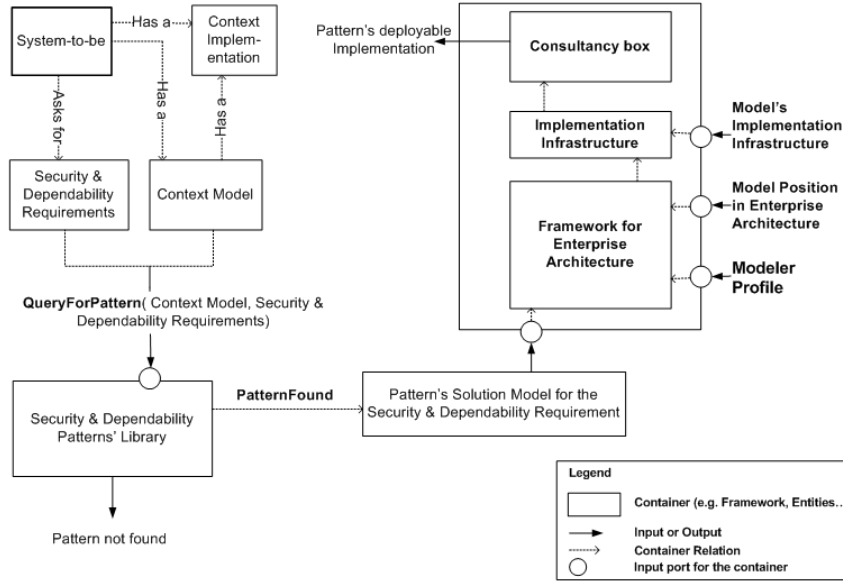


Fig. 3. Methodology for Deployment of S&D Organizational Patterns

Enterprise Architecture defines (and extends) the S&D patterns with an enterprise type of context semantics. The basic idea is to deduce the deployment process of patterns by answering the questions ‘What is the role of the pattern requester in the organization (i.e., what is his/her interest and duties in the organization)?’ and ‘In which aspect its S&D requirement is needed?’. The retrieved answers are input \langle **MODELER PROFILE**, **MODEL POSITION IN THE ENTERPRISE ARCHITECTURE** \rangle in the Zachman framework to retrieve the context specifications. This makes it possible to infer the unique cell that provides semantic information about the context in which the organizational S&D pattern is implemented. An example of how to browse the Zachman framework has been presented in Example 3.2. Nevertheless, this acquired context remains independent from the type of infrastructure.

In Section 3.3, we have discussed four types on implementation infrastructures. These infrastructures enrich organizational patterns by anticipating the actual implementation infrastructure of the system-to-be. Not any type of infrastructure may be suitable for all cells of the Zachman Framework. However, we leave to experts the task of associating deployable solutions to limit the “Enterprise Architecture \mapsto Implementation Infrastructure path” choices.

Using the gathered information, the consultancy box provides the requester with a set of interpretations described through experts’ analysis. The consultancy box’s conclusion is based on the information provided by requesters through the three inputs. Specifically, the last input specifying the implementation infrastructure of the system-to-be splits the consultancy proposals into either *recommendations* (follow-up for the parameter corresponding to Software or Hardware implementation infrastructures) or *guidelines* (follow-up for the parameter corresponding to Policy and Human Resource implementation infrastructures).

4.1 Towards Deployment Recommendations for Organizational Patterns

All along the methodology, a more detailed deployment context has been identified. The Software or Hardware implementation infrastructures, obviously, can be driven the definition of S&D solutions (e.g., implementation code) that are more accurate than those provided by the pattern library.

Example 4.1 *The requester (i.e., sub-contractor) wants to ensure high reliability of the healthcare system (Req 7). Nevertheless, the pattern solution described in Example 3.1 requires the duplication of unreliable services (e.g., the web service managing operations on the EHR). This, however, is not an appropriate interpretation. Previously, the pattern requester selected his requirements constraining the operational deployment of the solution. Currently, the requester selects the software implementation infrastructure as the where the solution has to be applied. The reliability requirement in addition to the new enhanced context are used to fetch appropriate candidate S&D patterns specific for software implementation infrastructure. These new patterns are retrieved from the SERENITY library directly.*

4.2 Towards Deployment Guidelines for Organizational Patterns

Guidelines are follow-up consultancy for the management of the expertise captured in the organizational pattern for Policy and Human Resource implementation infrastructures. The deployment of S&D organizational patterns into the system means to provide the system designer with the list of modifications needed to be applied before starting the actual pattern implementation. Since our objective is to assist system designers during system development, the pattern solution should be described together with a group of guidelines for possible (including combined) infrastructures in order to match the real needs of the organization and facilitate the reuse of existing infrastructures.

Once an S&D pattern is selected, its deployment requires the system designer to take the following actions:

- (i) *instantiate the general terminology of the pattern to the one of the system.*
During this step, analysts should understand the correspondence between the pattern and the concrete instance of the system. Essentially, they have to identify the correspondence between the general terms (i.e., actor-name, goal-name, etc.) used in the pattern, and the actual instances within the system. In case analysts disagree or cannot define these correspondences, the pattern instantiation fails and the deployment process is halted.
- (ii) *confirm preconditions that can not be verified automatically by the pattern.*
Most of the S&D patterns constrain their deployment to other conditions. Preconditions are conditions that need to be available before pattern deployment. The success of the deployment directly relates to them. Before selecting the pattern, preconditions are checked by analysts. Preconditions at the level of deployment are infrastructure-orientated. They generate confirmation questions about organizational infrastructure conditions (e.g., Do you confirm that there is enough budget to higher one part time employee?).
- (iii) *modify the system corresponding to the guidelines provided by the pattern de-*

ployment. The guidelines, presented in a document, describe the modifications required for possible specific infrastructure(s) of the solution. Usually, this description is provided to the software designer in a structured natural language format.

The guidelines provided by the pattern deployment process are defined by the security experts. These expert suggestions correspond to the guidelines they usually propose during consultancy. This totally captures these experts consultancy during pattern deployment.

Example 4.2 *The Redundancy pattern has been proposed to guarantee the compliance of the system with Req 3 in Table 1. Previously, the pattern requester has defined his requirements towards an operational deployment of the solution (see Example 3.2). In particular, the ‘Builder’ has chosen the Human Resource infrastructure for the deployment of this solution. Following the guidelines described above, the system designer has to instantiate the context of the Redundancy pattern using the terminology used in the remote healthcare system domain. For instance, the mapping assigns “Alert Calls” into unreliable services and the “MERC employee” to the person in charge of receiving those requests. Once the correspondence between the pattern context and the application domain is established, pattern preconditions are verified. The ‘Builder’ needs to understand if the Human Resource infrastructure of the MERC can accommodate new receivers. The guidelines suggest to hire a new employee and grants him the rights to receive, view and respond to the alert requests. The pattern solution in the SERENITY library describes reliable services as those services for which more than one actor are in charge of their achievement. Thereby, the last guideline modifies the alert workflow performed with the MERC by ensuring that both employees – the initial employee and the new employee – participate to the workflow.*

5 Conclusion

This paper has presented a methodology for the deployment of S&D organizational patterns. It relies on inputs from the pattern requester for selecting (sets of) deployable solutions. The first two inputs are fed to the Zachman Framework to extend the pattern with context from the Enterprise Architecture. The third input specifies the implementation infrastructure of these systems. The consultancy box is an analytical operation applied to the pattern solutions already enriched with information about the system. It stores guidelines and recommendations of experts about available patterns. *Guidelines* are used to describe solutions for Human Resource and Policy implementation infrastructure. *Recommendations* concretize the deployment with refined software and/or hardware related patterns.

The Zachman Framework implementation using Protégé is part of on-going work at University of California Irvine.⁹ Still one deficiency in our approach is in validating that requesters correctly followed the guidelines proposed by the methodology. The usual software testing approaches (such as model-based testing) allow technical investigation of the properties available in the product with respect to their

⁹ Available online at <http://apps.adcom.uci.edu/EnterpriseArch/Protege/Zachman.html>

intended operational procedure. Intuitively, we cannot apply such approaches to validate the compliance between the S&D patterns and the pattern deployment. As future work we aim to describe an approach that verifies whether the system designer succeeds or fails in following these deployment guidelines by monitoring of system behavior or by monitoring both system and deployed pattern behavior in case where the pattern has been implemented using software and/or hardware infrastructures.

Acknowledgments

This work was partially funded by the IST-FP6-IP-SERENITY. We thank all members of the SERENITY project for their feedback and useful scientific discussions.

References

- [1] Anderson, R., *Why cryptosystems fail*, Communications of the ACM **37** (1994), pp. 32–40.
- [2] Basin, D., J. Doser and T. Lodderstedt, *Model Driven Security: from UML Models to Access Control Infrastructures*, ACM Transactions on Software Engineering and Methodology **15** (2006), pp. 39–91.
- [3] Compagna, L., P. E. Khoury, F. Massacci, R. Thomas and N. Zannone, *How to capture, communicate, model, and verify the knowledge of legal, security, and privacy experts: a pattern-based approach*, in: *Proc. of ICAIL'07* (2007), pp. 149–154.
- [4] Fusaro, P. C. and R. M. Miller, “What Went Wrong at Enron: Everyone’s Guide to the Largest Bankruptcy in U.S. History,” Wiley, 2002.
- [5] Gamma, E., R. Helm, R. Johnson and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley, 1994.
- [6] Giorgini, P., F. Massacci and N. Zannone, *Security and Trust Requirements Engineering*, in: *Foundations of Security Analysis and Design III*, LNCS 3655, Springer, 2005 pp. 237–272.
- [7] Hurst, A., R. Channing and P. Bolding, *SocGen trader faced earlier inquiry: report*, Reuter (February 18, 2008).
- [8] Jafari, R., F. Dabiri, P. Brisk and M. Sarrafzadeh, *Adaptive and fault tolerant medical vest for life-critical medical monitoring*, in: *Proc. of SAC'05* (2005), pp. 272–279.
- [9] Johnston, D., *Russian accused of citibank computer fraud*, The New York Times (August 18, 2007).
- [10] Jürjens, J., “Secure Systems Development with UML,” Springer, 2004.
- [11] Kim, T.-H. and H. yeol Kwon, *Applying Security Engineering to Build Security Countermeasures: An Introduction*, in: *Proc. of PARA'04*, LNCS 3732 (2004), pp. 957–963.
- [12] Konrad, S., B. Cheng, L. Campbell and R. Wassermann, *Using Security Patterns to Model and Analyze Security Requirements*, in: *Proc. of RHAS'03*, 2003.
- [13] Massacci, F., J. Mylopoulos and N. Zannone, *An Ontology for Secure Socio-Technical Systems*, in: *Handbook of Ontologies for Business Interaction*, The IDEA Group, 2007 .
- [14] Nuseibeh, B. and S. Easterbrook, *Requirements Engineering: a Roadmap*, in: *Proc. of ICSE'00* (2000), pp. 35–46.
- [15] Pigot, H., A. Mayers and S. Giroux, *The intelligent habitat and everyday life activity support*, in: *Proc. of the 5th Int. Conf. on Simulations in Biomedicine*, 2003, pp. 507–516.
- [16] Schumacher, M., “Security Engineering with Patterns: Origins, Theoretical Models, and New Applications,” Springer, 2003.
- [17] Yoder, J. and J. Barcalow, *Architectural Patterns for Enabling Application Security*, in: *Proc of PLoP'97*, 1997.
- [18] Zachman, J. A., *A Framework for Information Systems Architecture*, IBM Systems Journal **26** (1987), pp. 276–292.