Formalization of security patterns as a means to infer security controls in business processes

ANGEL JESUS VARELA-VACA* and RAFAEL M. GASCA, *Department of Computer Languages and Information Systems, University of Seville. Escuela Superior de Ingeniería Informática, Avd. Reina Mercedes s/n, Seville, Spain*

Abstract

The growing trend towards the automation and externalization of business processes by means of Technology Infrastructure (TI), such as Business Process Management Systems, has increased the security risks in the organizations. In the majority of cases, the issue of security is overlooked by default in these systems. Therefore, the early selection and implementation of security controls that mitigate risks is a real and crucial need. Nevertheless, there exists an enormous range of IT security controls and their configuration is a human, manual, time-consuming and error-prone task. In addition, security controls are implemented out separately from the organization perspective and involve many stakeholders. This separation makes difficult to ensure the effectiveness of these controls with regard to organizational requirements. In this article, we propose a formalization of security controls based on security pattern templates and feature models. This formalization allows applying feature domain-oriented analysis and constraint programming techniques for the automatic inference, selection and generation of optimal security controls with regard to single and multiple business objectives.

Keywords: Business process, security patterns, feature model, constraint programming, optimization.

1 Introduction

There exist a growing trend towards the automation and externalization of business processes by means of Technology Infrastructure (TI), such as Business Process Management Systems (BPMS), has increased the security risks in the organizations. Nevertheless, security in BPMS is mostly overlooked by default or is taken into consideration in a second phase. It is crucial for organization to protect the control flow and data flow to their business processes against security risks. Since the cost and consequences of the materialization of risks in these systems could range from mildly annoying to catastrophic, with serious injury occurring or systems destroyed, reputation losses, security breaches and so on. In general, there is a lack of mitigation policies or risk treatment plans within organizations. It is therefore crucial to act as soon as possible in selecting, developing and monitoring adequate security controls that mitigate or reduce the consequences of these risks.

The selection and configuration of security controls is one of the main problems within the scope of IT security since, in most cases, it constitutes a human, manual, time-consuming, and error-prone task that involves several security stakeholders, such as security managers and administrators [10]. Ideally, this task should be automated to reduce the workload for security stakeholders, and to increase the profit of organizations. In [23], a report states that the automation of security controls will substantially reduce the cost of security, while improving its effectiveness.

There exist an enormous number of catalogues, such as those defined by COBIT, COSO, ISO/IEC 27002:2013, HIPAA and PCI DSS 3.0, which can vary from abstract to specific controls depending

*E-mail: ajvarela@us.es; gasca@us.es

on the level of the specification to be implemented. Traditionally, security controls correspond to ad hoc configurations due to late and unexpected threats. Security controls can vary from the simple installation of software (e.g. anti-virus software) to the configuration of secure protocols (e.g. https protocol) or even the configuration of an Access Controls List (ACL) for a network firewall. This heterogeneity coupled with the absence of formalism and a huge variability complicate the task of selection and configuration of specific security controls since it implies the involvement of a high level of knowledge and expertise.

To overcome these problems, we propose a formal model for the representation of security controls based on security patterns. Security patterns have been enhanced by means of a formalization based on feature models (FMs). This formalization enables the reasoning capabilities, such as checking the effectiveness and suitability of security control configurations with regard to the needs of the organizations. The automated analysis of security patterns by means of Feature-Oriented Domain Analysis (FODA) enables the reasoning capabilities in the form of the selection of components, selection and derivation of parameter values, layout of the selected components, etc. Consequently, Constraint Programming (CP) techniques based on optimized and non-optimized searches have been applied to automate FODA for the selection and generation of configurations.

The article is structured as follows: Section 3 gives an introduction to feature-oriented model analysis concepts; Section 2 presents an overview of related work found in the literature; Section 4 presents the proposal modelling and formalization of security patterns; Section 5 presents a case study where a security pattern is formalized and the results of analysis are shown; and in Section 7 conclusions are drawn.

2 Related work

The optimal selection of security controls and configurations has been tackled in two previous studies [31, 32]. In [31], the optimal selection of configurations is formalized as an optimization problem using FODA and CP techniques. Additionally, a catalogue of FMs is presented to enhance various security objectives for BPMS. In [32], a security pattern-based approach for the formalization of security controls is presented as a risk treatment plan in business processes.

There exist several contributions where the selection of risk mitigation plans at the *Design and Analysis* phase of Business Process Management lifecycle is treated. In [14], Risk-Oriented Process Evaluation (ROPE) method is proposed. ROPE includes a *Counter Measure Sub-Process* where countermeasure actions are defined against threat activities. Nevertheless, this approach lacks of proposal for the automatic and optimal selection of suitable countermeasures. In [7], the authors propose a phase in the proposal for risk mitigation: however there is no well-accepted formal theory that describes how to carry out this task. In [2], the authors provide a goal-driven approach as an extension of Tropos/i* to analyse risk at organization level. Furthermore, they illustrate a number of different techniques to help the analyst in identifying and enumerating relevant countermeasures for risk mitigation. Nevertheless, the authors focus on studying the selection of countermeasures in cost-effective terms. In [12], the authors propose the selection of ISO/IEC 27001 controls based on multiple objectives, (cost, benefit, etc.). Nevertheless, it is merely indicated that selection is carried out using search-based techniques, and there is no reference to implementations. In [3], a probabilistic framework is presented for risk assessment, and an optimization-based approach is proposed to determine the manner in which control procedures can be embedded in the business process to mitigate risk. Nevertheless, it is a theoretical approach that fails to indicate how the optimized selection of controls are carried out. On the other hand, automatic techniques are proposed

in [8] to determine mitigation actions at run time of YAWL business processes. These mitigations are determined by MOSA algorithms that enable the selection of countermeasures by using multiple objectives.

Regarding to security patterns, there exist relevant approaches where security patterns has been formalized such as [18, 25, 26]. The formalization in [25] is focused on the definition of ontologies to map security concepts within security patterns to enable search engines and query capabilities. In [26], a catalogue of security patterns is defined using natural language and Unified Modeling Language (UML) diagrams. Nevertheless, these formalizations are unsuitable to automate the application of security patterns. In [18], security patterns have been formalized as profiles to automate the generation of security policies for Service Oriented Architecture (SOA) environments. Menzel provides different security patterns to fulfil certain security intentions such as *User Authentication*, *Identity Provisioning*, *Data Confidentiality* and *Data Authenticity*. Furthermore, security patterns has been formalized by means of ontologies and pseudo-formal grammar. The main problem in the approach is the coupling to the solution. That is, security model, formalization and security patterns have been defined to support the specification of solutions only valid to SOA environments.

There exist certain studies in the revised literature with respect to FODA and the generation of models. In [27], an extension of a goal-driven method (KAOS) is proposed to generate adaptive requirement models from variant models. In [16], an approach that facilitates the development of secure software product lines (SPLs) and their derived products is proposed. In [22], FMs are used to analyse the variability requirements and consequently transform this FMs in order to generate an architectural model. Nevertheless, these approaches are focused on the generation of requirement models or of products related to software, while our approach strives to determine the configuration through FMs. In [24], the authors use FODA to provide self-adaptive systems by dynamically determining the best variants suited to specific QoS requirements. The authors use a goal model extended with attributes (characteristics of features) and soft goals (QoS requirements) in order to represent the FM, and by means of FODA, they determine which configuration is suitable for each context; thus values for goals and soft goals are required. The main drawback in these approaches is that they use qualitative domains for attributes in order to determine variants as a consequence of using logic programming. In contrast, quantitative domains for attributes are supported in our approach.

3 FODA in a nutshell

Software Product Line (hereinafter SPL) [4], is an emerging paradigm in the Software Engineering arena which provides guides for the development of products. This paradigm is based on the key idea of defining reusable components by means of the identification of core features and through product development. Current studies in SPL are focused on the domain analysis that consists of the process of analysing related products in order to identify their common and variable features. FM is the most popular method for the domain analysis of a SPL. FODA [15] is a method to perform a domain analysis of FMs.

FMs involve a model that defines features and their relations. FMs enable to study certain properties such as potential number of valid products, even whether a particular configuration (selection of features) constitutes a valid product. There exist various notations to design FMs, although the most widely used is that proposed by Czarnecki [9] as shown in Figure 1. This representation enables four relations between a parent feature and its child features such as shown in Figure 1.

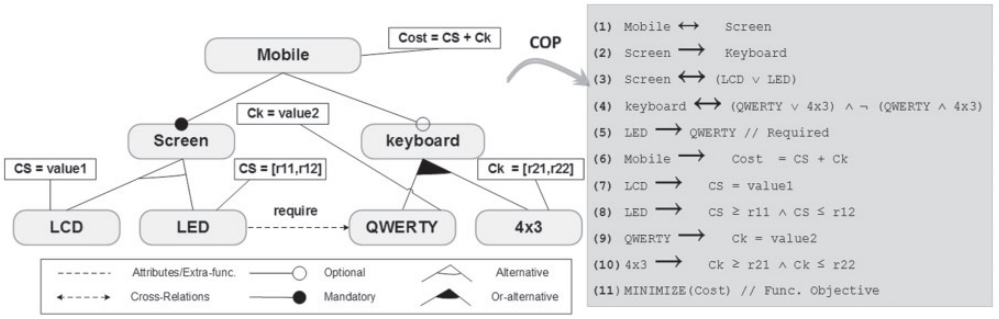

FIG. 1. Example of a FM and its translation to a COP.

In some cases, the expressiveness of this notation is insufficient to represent certain relations and information related to features. To overcome this drawback there exist extensions [9] for the inclusion of attributes and extra-functionalities for features. These extensions enable characteristics of features that can be measured to be provided and include the facility to express relations between these characteristics (extra-functionalities). Figure 1 shows an example of an FM extended with attributes for instance feature *LCD* has an attribute *CS* which is instantiated by the expression $CS = value1$.

Several techniques are available for the automated analysis of FMs [5], using Propositional Logic (PL), CP and Description Logic (DL). These approaches transform the FMs into formal models in order to infer information related to the product line. This information may include: number of products, filters (specific set of characteristics for the features), products (all products with certain features), validation (selection of characteristics represent a valid product), optimum products (determination of best products according to a set of criteria), variability (relation between number of potential products and certain products), commodity (relation between a number of certain products and the total number of products).

CP is employed to carry out FM analysis since this approach enables integer domains to be used for attributes and optimization functions. In [6], the authors apply a transformation to Constraint Optimization Problem (COP) of extended FMs which is able to automatically obtain information about the model. Figure 1 gives an example of transformation of an FM to a COP whose objective function is to find the minimum of *Cost*.

As previously introduced, one of the main problems in risk treatment is how to describe and/or model controls. We propose to employ security patterns in the representation and modelling of security controls since security patterns [26] are a widely recognized means for the description of security solutions. Nevertheless, security patterns are usually defined in a textual, informal way, using natural language. In the following section, basic concepts of security patterns are introduced and a formalization based on FMs of security patterns is given.

4 Formalization of security controls

Security patterns are based on the idea of design patterns that were introduced by Christopher Alexander in 1977: *A pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that pattern*.

In general, security patterns, as defined by Schumacher *et al.*, are described in documents that have a specific structure:

- *Name* identifies and summarizes the pattern.
- *Context* describes the environment before the application of this pattern.
- *Forces* are conditions within the context.
- *Problem* describes a problem that occurs within the context.
- *Solution* describes how to solve problems within the context.

These constitute the mandatory elements; however, other sections, such as dependencies with other patterns, implementation details, structure, examples, can be incorporated into the security patterns in order to improve the information.

In general, security patterns [25] are defined in an informal way, using the natural language. Microsoft provides a library of patterns for Security in Web Services [19]. Nevertheless, these patterns are presented in a textual and informal way. In [20], a catalogue of security patterns is defined for security identity and service provisioning in J2EE applications. These patterns are focused on the implementation aspects of patterns. The formalization of the security pattern has been addressed in [25]. The authors propose the definition of a knowledge database using security pattern ontologies that enable security pattern concepts and security concepts to be linked. This approach strives to enable search engines and query capabilities. In [26], a catalogue of security patterns is defined using natural language and UML diagrams. However, these formalizations are unsuitable for the automation of the application of security patterns. On the other hand, in [18], security patterns have been formalized as profiles to automate the generation of security policies for SOA environments. Menzel provides various security patterns in order to fulfil certain security intentions such as *User Authentication Identity Provisioning*, *Data Confidentiality* and *Data Authenticity*. Furthermore, these security patterns have been formalized by means of ontologies and pseudo-formal grammar.

In our approach, the security patterns are utilized for the selection and generation of security configurations as security controls for certain vulnerabilities. Hence, the proposed security pattern model must introduce new capabilities with concepts and formalisms for the enhancement of inference in an automatic way. We propose a security pattern meta-model, such as that shown in Figure 2, whereby security pattern concerns have been related with the risk model described in previous work [32]. The risk model is an extension that integrates business objectives (Business Motivation Model), business process model (Business Process Meta-Model) and risks (UML Profile for QoS and FT) into the same meta-model. This risk model provides an extension for business processes according to the concepts of the UML Profile for Modelling Quality of Service (QoS) and Fault Tolerance (FT) Characteristics and Mechanism Specification, and BMM. The risk model provides the basic entities, relations, additional artefacts and properties necessary to carry out a risk assessment in business processes.

This extension strives to represent controls (cf. countermeasure in Figure 2) as security patterns. Similarly, the security pattern meta-model has been related to the risk model by means of dotted-relations. There are two main relations: (1) *Problem* is related with vulnerabilities of the risk model; and (2) *Security goals* and intentions established by a security pattern are related to business objectives defined in the business process through the risk model. For a better understanding of how security patterns are formalized through the proposal meta-model, its concepts and relations are detailed below.

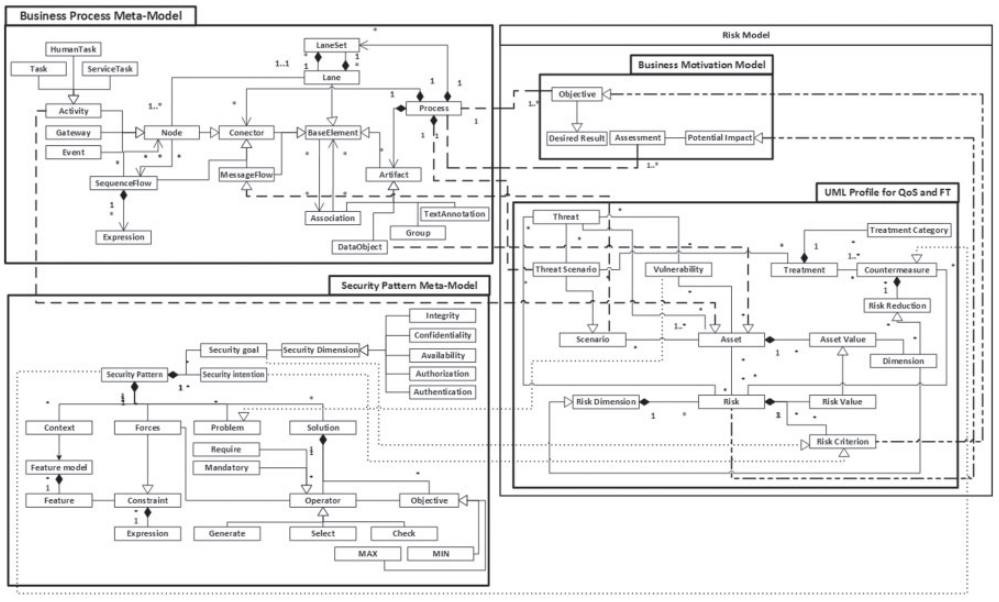

FIG. 2. Security pattern meta-model for business processes.

4.1 Context

As mentioned earlier, *Context* describes the environment and its properties before the application of the pattern. We propose formalizing the *Context* by means of the specification of FMs. Thus, FMs are not only able to define the components within the context, the supported features for the components and their relations, but it also is possible to represent the configurations for the different features of the context. In general, FMs are described in a graphical way; however, constraint languages can be used as a formalism to represent the FM, such as that shown in Section 3. The analysis of FMs also give inference capabilities that enable extra information to be attained. In [31], a catalogue of five-FMs have been formalized to enforce the security dimensions of confidentiality, integrity, authentication and availability in BPMS.

In [18], a description of *Context* is disregarded within security patterns. Nevertheless, the author uses ontologies and diagrams as support in the description of *Context*. For instance, an ontology for *Identity Management* is given to describe which kind of tokens can be configured as credentials. Each security pattern then defines a diagram to show how the components within the context are linked. For instance, *Broken Authentication* pattern uses four components: a client, two identity providers Secure Token Service (STS), and a service.

A context example is shown in Table 1 where there is an example of a constraint that indicates a condition for protocol *SSLv3*.

4.2 Problem

As aforementioned, *Problem* describes the needs and situations that are produced within *Context* to provide a solution. We propose using identifiers established by CWE [1] for *Problem*

TABLE 1. Summary of security pattern elements

Name	Type	Example
Context	Constraints	$Protocol = SSLv3 \rightarrow ProcotolCost \leq 300$
Security goals	Feature token	*Integrity*
Security intention	Feature token	*Data integrity*
Problem	Feature token	*CWE-523: Unprotected Transport of Credentials*
Forces	Constraints	$OPTIONAL(Procotol)$
Solution	Constraints	$SELECT(MIN(ProtocolCost))$

since it provides one of the most relevant and public-access vulnerability database. CWE is currently supported by public and private institutions, such as IBM, Hewlett-Packard, SANS Institute and Information-technology Promotion Agency, Japan (IPA), etc. In Table 1, there is an example of security pattern which *Problem* is specified using CWE-89 referred to SQL injections vulnerability.

For instance, *Direct Authentication* [19] is a pattern defined by Microsoft where the *Problem* is described by a sentence such as 'How does the Web service verify the credentials that are presented by the client?' In this particular case, the Problem is that there exists a need of verification of authenticity of clients in Web Services. This pattern proposes a solution based on direct authentication, that is, the Web service acts as an authentication service to validate credentials from the client. Nevertheless, these descriptions are very textual with a lack of formalization. We therefore propose formalizing *Problem* through concerns of threats and vulnerabilities.

There exist various vulnerability and threat databases, such as Common Weakness Enumeration (CWE) [1], and NIST Vulnerability Database [21], which provide information referring to technological vulnerabilities. Identifiers utilized by CWE or NIST can be adopted for the specification of security patterns in our approach. For instance, *CWE-89* is used to refer to a kind of SQL injection vulnerability in CWE. In addition, CWE provides a particular section (Common Consequences) to indicate which security goal is affected. In the same way, other databases, such as CRAMM database can be adopted by means of defining the alignment of its concepts.

4.3 Security goals and intentions

We propose the inclusion of new descriptors for *Security goal* and *Security Intention* concerns in order to specify new criteria for the selection and inference in security patterns. *Security goal* is used to refer to a security dimension (e.g. integrity, confidentiality, availability, etc.) that the security pattern is enforcing. Security intention is used to describe those aspects related to the security goals. In Table 1, there is an example of security goals related to integrity and a security intention that indicates that the pattern strives towards integrity in data.

In other approaches, such as those approaches in [18] and [17], the authors define a relation with security intentions. For instance, in [18], the authors define the security patterns: *Secure Pipe* and *Information Protection* as intentions. However, it is not clear which security goals the pattern is enforcing, since, a security intention could enforce various security goals, and therefore this relation is also fuzzy.

4.4 Forces

Forces are conditions and duties within the *Context* that are necessary or mandatory for the application of the pattern. Therefore, *Forces* can be defined by constraints classified into two categories: (1) mandatory requirements; and (2) optional requirements. For instance, *Direct Authentication* pattern in [19] defines force such as: 'The Web service can validate credentials from the client against an identity store.' In this example, there is a mandatory requirement that indicates that an identity store is required and that there should exist a direct channel of communication with Web services. In [18], *Forces* is defined by a small grammar with two operators: FORALL and ASSERT. These operators are used to enforce certain conditions with regard to objects of the model.

We propose to formalizing *Forces* with constraints to represent mandatory (MANDATORY) and optional (OPTIONAL) conditions. Thus, this formalization will enable us to indicate when a feature and a configuration of features must be mandatory or optional. For instance, OPTIONAL (*IdentityStore*) could represent a constraint which indicates that a feature of an identity store is required. The proposal formalization can be described in the form of Backus-Naur Form (BNF) grammar as follows:

$$ForcesExpr ::= Expr \quad OP \quad Expr \tag{4.1}$$

$$| \quad Expr \quad | \quad \neg Expr \tag{4.2}$$

$$OP ::= BoolOp \quad | \quad RelationOp \quad | \quad MathOp \tag{4.3}$$

$$BoolOp ::= \vee \quad | \quad \wedge \tag{4.4}$$

$$Expr ::= ForcesExpr \quad | \quad val \tag{4.5}$$

$$| \quad OPTIONAL(Expr) \quad | \quad MANDATORY(Expr) \tag{4.6}$$

$$RelationOp ::= \quad < \quad | \quad \leq \quad | \quad = \quad | \quad > \quad | \quad \geq \quad | \neq \tag{4.7}$$

$$MathOp ::= + \quad | \quad - \quad | \quad \times \quad | \quad \div \tag{4.8}$$

Example of Table 1 shows *Forces* that indicate that *Protocol* is an optional feature. Thus, the specification of protocol in the security configuration is optional.

4.5 Solution

Solution should represent mechanisms for the solution of a *Problem*. For instance, *Solution* is given in a textual description in *Direct Authorization* example in [19]. On the other hand, Menzel uses a grammar similar to that used in *Forces*. In that case, Menzel introduces a set of operators such as: REQUIRE, ENFORCE, SET, USE, SCOPE. These operators are used to describe *Context* configuration.

For our proposal, *Solution* aims to find the optimal security configuration for a *Problem*. Thus, the solution is to find a valid configuration within a *Context*. Therefore, we propose to formalize *Solution* using two operations: (1) *SELECT(obj)* indicates that the goal of the pattern must be the selection of all valid configurations according to the constraints established in *Context* and *Forces*; and (2)*GENERATE(obj)* indicates that the goal of the pattern must be the generation of a configuration according to the constraints established in *Context* and *Forces*, where *obj* represents a list of objectives in both operations. Objective functions are represented by MAX and MIN functions. We propose to use a representation as the following: *MAX(obj)*, indicates that the objective function is to maximize the value of *obj*, being *obj* a metric or a function; and *MIN(obj)*, indicates that the objective function is to maximize the value of *obj*, being *obj* a metric or a function.

In case of the multiple objectives, a list of objectives can be stated. For example, whether the objective selects a configuration such as minimize the $metric_1$ and maximize $metric_2$. This operation can be stated as follows: SELECT(MIN($metric_1$),MAX($metric_2$)).

The formalization of the security patterns proposed involve at least two different stakeholders: business and security analysts. On the one hand, business and security analysts must define the domain of the problem (cf. *Context* section) as an FM. This task requires great effort in the analysis of the current system, software, tools, procedures, etc. On the other hand, *Problem*, *Security goal* and *Security intention* are predefined tokens used to characterize the pattern. These sections can be obtained automatically from known databases, such as CVE, CWE and NIST. Other requirements and objectives (cf. section 4.4 and sections 4.5) are constraints imposed by the organizational needs. These constraints must be defined by the business and security experts to complement the information of the pattern with regard to requirements of the organization.

In next section, we present a case study where a security pattern for confidentiality, integrity and authentication is presented. The formalization has been supported by the FM presented in [31]. Subsequently, we have carried out an analysis of the security pattern in order to infer certain configurations with regard to a single and multiple objectives.

5 A case study: security pattern for confidentiality, integrity and authentication

This case study has been applied to ensure secure communications in BPMS within research projects that have been successfully certified by the standard ISO/IEC 27001 Information technology—Security techniques—Information security management systems—Requirements [13]. Further case studies are included in previous results in [30], such as the enhancement of authentication in BPMS and the provision of availability and integrity in BPMS.

In real scenarios there exist channels of communication between business processes with other systems. CWE [1] describes a vulnerability; *CWE-523: Unprotected Transport of Credentials*, referring to the application of bad measures to protect credentials in a communication channel. To ensure the integrity and confidentiality of information exchanged in these communications, a secure channel is mandatory. A secure channel requires the application of digital signatures and encryption infrastructure. This infrastructure can be applied at the transport (SSL/TLS) or message (WS-Security) layer. Secure Socket Layer (SSL) protocol and Transport Layer Security (TLS) protocol [28], (hereinafter SSL/TLS) are widely used to provide confidentiality, authentication, and integrity in data communications. In fact, CWE propose the enforcement of SSL in the transport layer.

Most application servers for common web applications employed to deploy business processes, such as *Oracle WebLogic, IBM WebSphere, Apache Tomcat and JBoss*, support the establishment of secure channels by means of SSL/TLS. Figure 3 shows an example of a configuration of an SSL Socket connector for a *Jetty* server to use a specific CipherSuite. Other servers, such as *Apache Tomcat and JBoss*, enable this type of configuration in a similar way.

The configuration of SSL/TLS on these servers is based on the establishment of features such as certificates, certificate authorities, key-stores, ciphers, ports and protocol versions. Following the standards, connections negotiate a CipherSuite to be used in the communication. *CipherSuite* presents an enormous combination of configurations since each *Key Change Method* can be combined with a number of *Ciphers* and *MAC* algorithms. Figure 4 shows the resultant FM of the analysis carried out for the current version of the standards and the configuration of *SSL/TLS* for *Oracle WebLogic, IBM WebSphere, Apache Tomcat and JBoss using OpenSSL* and *JSSE* providers.

```
<New class="org.mortbay.jetty.security.SslSocketConnector">
  <Set name="Port">8443</Set>
   ...
  <Set name="IncludeCipherSuites">
    <Array type="java.lang.String">
      <Item>TLS_DHE_DSS_WITH_AES_256_CBC_SHA</Item>
      <Item>TLS_DHE_RSA_WITH_AES_256_CBC_SHA</Item>
      ...
    </Array>
```

FIG. 3. Example of SSL/TLS configuration for a Jetty Server.

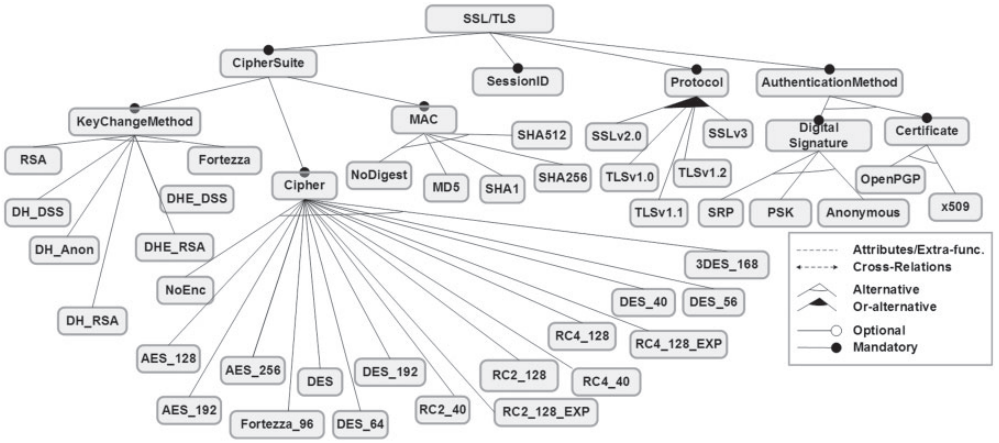

FIG. 4. FM of SSL/TLS [31].

The FM only shows the values that are supported for each feature, although numerous constraints interrelate these features. Due to the lack of expressivity of FMs, this kind of constraint can be represented as cross-tree constraints. For instance, SSL v3.0 introduces Fortezza as a Key Exchange Method, although this method cannot be used with MD5 as a Message Authentication method (MAC).

In addition, various metrics, such as Annual Loss Exposure (ALE), Annual Rate of Occurrence (ARO), Reduced Annual Rate of Occurrence (RARO), Single Loss Exposure (SLE), ROSI (Return of Security Investment) and Cost, typically used to measure the cost–benefit of security, have been used to extend FM functionalities, as shown in [31]. The relation metric-features are defined using extra functions. A set of extra functions (functions and constraints) are included in the FM that are listed together the formalization of the FM in [31].

These metrics enable the suitability of the solution in terms of cost–benefit to be measured. The organization could use the security pattern defined and state a search for a configuration with the minimum *ALE*, minimum *Cost*, or the highest *ROSI* as possible. This search can be specified using an statement for *Solution* such as *SELECT(MIN(ROSI))*.

TABLE 2. Security pattern for the enforcement of SSL in BPMS [31]

Name	Description
Context	*FeatureModel*^{c,i,at}
Problem	*CWE-523: Unprotected Transport of Credentials*
Security goals	*Confidentiality, Integrity, Authentication*
Security intention	*Enfocerment SSL/TLS*

TABLE 3. Results of FODA

NF	Mandat.	Optional	XOR	Or	Void	Number of conf.	Time (ms)
49	10	0	42	5	×	3.683	4.699

Finally, we have formalized a security pattern as shown in Table 2 in order to provide solutions to the vulnerabilities *CWE-523* by means of the enforcement of SSL in the transport layer of communication channels. The use of SSL ensure the goals of confidentiality, integrity and authentication. The FM in Figure 4 has been used as *Context* (referred as *FeatureModel*^{i,c,at} of Table 2) for this security pattern.

5.1 Performance results

To automate the selection of configurations, FODA based on CP techniques can be used. FMs have therefore been implemented into Constraint Satisfaction Problems (CSP) following the approach [5]. FMs have been implemented in *COMET*©. *COMET*© is a very powerful constraint solver with features for optimized searches. Since SAT solvers [5] are more commonly employed in this kind of analysis due to their promising performance results achieved using the *COMET*© solver, we have decided to carry out the entire analysis using this CSP solver. The constraint programs used in this section are available for evaluation and downloading at [29].

The first analysis consists of the identification of the total number of configurations. The result for this analysis is given in Table 3. The analysis indicates: number of features (NF); relations (mandatory, optional, XOR and OR); void FM [5] is a model validation operation that indicates whether the FM is void (cf. indicated by •) or not (cf. indicated by ×); number of configurations; and time of performance to obtain the configurations. An FM is void if it represents no products. The reasons that may make an FM void are often related with a wrong usage of cross-tree constraints. For this analysis, the (exhaustive) default search provided by the constraint solver is applied.

The number of configurations represents the number of all valid configurations that are achieved with this FM. Here, security administrators have to select from among all these configurations; however, such a large number of configurations cannot be handled by humans, such as in the case of SSL/TLS (*FeatureModel*^{i,c,at}). In the worst cases, the number of configurations might be prioritized by means of ordering criteria. It should be borne in mind that the time required to determine the configurations is very low even in the worst case.

As mentioned earlier, FMs have been extended with attributes and extra functionalities in order to adjust the searches. In this case, extended FMs might be transformed into COPs. A COP searches for a solution in accordance with an optimization function. Thus, COP strives to find the best

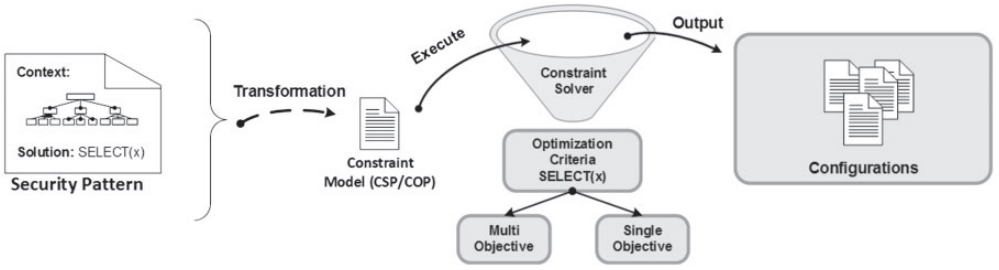

FIG. 5. Process of selection of configurations.

TABLE 4. Analysis of selection of configurations with attributes.

Optimization criteria	Number of conf.	Time (ms)	Constraint Model
SELECT(MIN(ALE))	13.138	2,041	COP
SELECT(MAX(RARO))	5.268	1,255	COP
SELECT(MIN(Cost))	1.800	2,394	MCOP
SELECT($\{MAX(RARO), MIN(ALE)\}$)	5.268	5,257	MCOP
SELECT($\{MIN(Cost), MIN(ALE)\}$)	0	406	MCOP
SELECT($\{\sim MIN(Cost), MIN(ALE)\}$)	108	880	MCOP

configuration feasible with an objective function with regard to the attributes included in the FM. For instance, the level of security of an organization can be considered high when the connections between client and server use certificates whose keys are generated using AES algorithms. These types of algorithms are only supported by TLSv1.0 and earlier versions. In this case, the organization can consider the attributes of security level, and is interested in configurations whose security level is high. Therefore, the COP aim would be to search for the best configuration with a high level of security.

In our approach, the generation of optimal configurations is proposed based on multiple objectives by means of Multi-objective COP (MCOP) [11]. In this case, COPs are adapted to attain multiple objectives. The transformation from FMs to COP and CSP have been carried out by adapting by hand the code of FMs to the search requirements. We have two options for the selection (cf. Figure 5): (i) one that provides all possible configurations (referred as *Determine all configurations SELECTION(x)*); and (ii) one that provides the best configurations (referred as *Determine optimal configuration SELECT(x)* according to certain optimization criteria (in the formalization denoted as F, and referred as *Optimization criteria MIN/MAX* in this figure).

To compare the results of the determination of configuration with and without optimization, a comparative analysis is given in Table 4, which assumes that the constraints and functions are included in the FMs. The comparative study has been performed in two phases: (i) a search using single objectives; (ii) a search using multi-objectives. The table shows information on the attributes included: the optimization function used for each case; the number of configurations achieved; the performance given in milliseconds; and which kind of constraint model has been used. The hardware employed to obtain the results in this section is an Intel Core i7 2,20GHz with 8 GB RAM and Windows 7 Home Premium OS.

As shown in Table 4, the *ALE* is calculated by the combination of *ARO*, *RARO* and *SLE*. In the cases of single objectives, the constraint solver finds all the configurations (if any). Nevertheless, in the case of multi-objectives, the constraint solver strives to identify the Pareto-efficient combinations. It can be observed that the first search retrieves no solutions in the case of multi-objectives. It is therefore impossible to find a solution that fits the minimum of *Cost* and *ALE* due to the problem being over-constrained. In this case, there are two options: (i) accept that there are no solutions; (ii) relax some of the objectives in order to find solutions close to the optimum. A multi-objective search for SSL/TLS ($FM^{i,c,at}$) has been performed, which relaxes the objectives (marked by the symbol $\sim$). In our initial tests, *SLE* and *ARO* hold a fixed value, and the *RARO* is calculated in terms of the configuration selected. In this case, it can be observed how the minimum has been tightened by eight iterations.

It should be pointed out that the number of configurations is high due to the combinations introduced by operators and the open domains in the FM. Table 4 shows the number of configurations achieved for each search. Configurations can help security stakeholders deal with decision-making regarding security configurations. This high number of configurations can be reduced through more refined searches that restrict the domains of attributes and add further search preferences. However, this high number of configurations indicates an overview of the domain of the problem (i.e. the space of configurations) and specific configurations can even be customized with respect to certain multi-objectives. However, such a large number of configurations could become inoperative for security administrators; as future research it should be interesting to introduce an ordering criterion in order to generate a list of the best configurations.

6 Discussion of the approach

The usefulness of the formalization can be observed from several perspectives of views:

- For business and security analysts, our approach enables the analysis and inference of security controls as configurations from security patterns.
- For security analysts, our approach enables them determine whether the configurations are more effective against certain requirements, constraints and objectives.
- For business and security analysts, our approach enables them to obtain blueprints or templates as security controls to configure systems. Furthermore, our approach can ensure that templates suit the organization requirements..
- In both cases, once security patterns are defined, they enable a quick, automatic, flexible and agile adaptation to many scenarios and requirements. For instance, business analysts can reuse the same Context for a variety of Problems. In addition, it is possible to customize the metrics for the same *Context* (e.g. a cost–performance metric) and objectives (e.g. infer a configuration that minimizes the cost against performance) in order to ensure the effectiveness of the configuration.

The major advantage of our approach from both points of view is that the time taken to complete a task such as the configuration of a system, which is typically time-consuming and error-prone, can be reduced to mere seconds. The main drawback of our approach is that it requires a high initial effort in the analysis of systems, tools and standards in order to provide a security pattern as complete as possible. Nevertheless, once security patterns are defined only update cycles are required.

7 Conclusion

The main obstacles in this BPMS are related to the lack of awareness in the IT security risk of business process-driven products. In the majority of cases, security is considered only as an afterthought. Selection and configuration of security countermeasures present a big challenge in BPMSs for several reasons: (i) it is a human, manual, time-consuming, and error-prone task; (ii) it involves many security stakeholders, such as business analysts, security managers and administrators; (iii) the selection of configurations according to multi-criteria requires very high expertise. We conclude that it is necessary to provide business and security stakeholders involved in the development of business process solutions with tools that aid the automatic selection of security configurations in accordance with the needs of the organization.

We propose the automation of the inference of security controls. To this end, we first provide a security pattern formalization that enables certain IT security controls to be modelled. Security patterns have been formalized by means of the definition of FMs. FMs ascertain which characteristics must support the BPMS in order to achieve certain security objectives. FODA and CP techniques have been proposed in order to automate the inference of configurations as security controls. This analysis enables, for example, the number of possible configurations to be ascertained, and also which configuration is the best option according to the attributes and functions included in the security pattern; it is even possible to ascertain whether a configuration is valid. CP techniques based on single and multi-objective searches help to improve the inference of configurations. Furthermore, these mechanisms provide a flexible and agile selection of configurations due to the easy-to-change security patterns by adding new forces or objectives. In a case study, the way to obtain the formalization of a security pattern for the enhancement of confidentiality, integrity and authentication is shown. Moreover, several searches have been applied using single- and multiple-objective strategies that show the number of available configurations and the performance obtained.

Acknowledgements

This work has been partially funded by the Department of Innovation, Science and Enterprise of the Regional Government of Andalusia project under grant P08-TIC-04095, by the Spanish Ministry of Science and Education project under grant TIN2009-13714, and by FEDER (under the ERDF Program). The authors would like to thank Dr Michael Menzel, Prof. Dr Christoph Meinel, Hasso-Plattner-Institute and FIDETIA Foundation for the valuable information that has contributed to develop the ideas in this article.

References

[1] Common Weakness Enumeration. http://cwe.mitre.org/index.html, 2011.

[2] Y. Asnar and P. Giorgini. Modelling risk and identifying countermeasures in organizations. In *Proceedings of 1st International Workshop on Critical Information Infrastructures Security (CRITIS'06). Volume 4347 of LNCS*, pp. 55–66. Springer, 2006.

[3] X. Bai, R. Padman and R. Krishnan. A risk management approach to business process design. In *ICIS*, p. 28, 2007.

[4] D. Batory. Feature models, grammars, and propositional formulas. In *Proceedings of the 9th international conference on Software Product Lines*, SPLC'05, pp. 7–20, Berlin, Heidelberg, 2005. Springer.

[5] D. Benavides, S. Segura and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: a literature review. *Information Systems*, **35**, 615–636, 2010.
[6] D. Benavides, P. Trinidad and A. Ruiz-Cortés. Automated reasoning on feature models. In *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAISE*. Springer, 2005.
[7] L. Churliov, D. Neiger, M. Rosemann and M. Z. Muehlen. Integrating risks in business process models with value focused process engineering. In *Proceedings of the 14th European Conference on Information Systems*, 2006.
[8] R. Conforti, A. H. M. ter Hofstede, M. La Rosa and M. J. Adams. Automated risk mitigation in business processes (extended version). March 2012.
[9] K. Czarnecki, S. Helsen and U. Eisenecker. Formalizing cardinality-based feature models and their specialization. In *Software Process: Improvement and Practice*, p. 2005, 2005.
[10] M. Davis. Research: 2012 strategic security survey, 2012.
[11] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization theory, methodology, and applications multiple criteria optimization: state of the art annotated bibliographic surveys. In M. Ehrgott and X. Gandibleux, (eds.) *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, vol. 52 of *International Series in Operations Research & Management Science*, ch. 8, pp. 369–444. Springer, 2003.
[12] S. Fenz, A. Ekelhart and T. Neubauer. Business process-based resource importance determination. In *Proceedings of the 7th International Conference on Business Process Management*, BPM '09, pp. 113–127, Berlin, Heidelberg, 2009. Springer.
[13] Foundation for Investigation and Development of Information Technologies (FIDETIA). ISO/IEC 27001 Certificate. http://www.fidetia.es/docs/Cert_27001.pdf, 2014.
[14] S. Jakoubi, S. Tjoa, G. Goluch and G. Quirchmayr. A survey of scientific approaches considering the integration of security and risk aspects into business process management. In *DEXA Workshops*, pp. 127–132, 2009.
[15] K. C. Kang. *Feature-oriented Domain Analysis (FODA): feasibility study; Technical Report CMU/SEI-90-TR-21 - ESD-90-TR-222*. Software Engineering Inst., Carnegie Mellon Univ., 1990.
[16] D. Mellado, E. Fernández-Medina and M. Piattini. Security requirements engineering framework for software product lines. *Information and Software Technology*, **52**, 1094–1117, 2010.
[17] M. Menzel, R. Warschofsky and C. Meinel. A pattern-driven generation of security policies for service-oriented architectures. In *Web Services (ICWS), 2010 IEEE International Conference on*, pp. 243–250, july 2010.
[18] M. Menzel. *Model-driven Security in Service-oriented Architectures*. PhD thesis, Hasso-Plattner - University of Potsdam, 2010.
[19] Microsoft. Microsoft patterns and practise proven practise for predictable results, 2012.
[20] R. Nagappan. Security patterns for j2ee applications, web services, identity management, and service provisioning, 2012.
[21] National Institute of Standards and Technology. NIST National Vulnerability Database. http://nvd.nist.gov/, 2011.
[22] J. Pérez, M. A. Laguna, Y. C. González-Carvajal and B. González-Baixauli. Requirements variability support through mda and graph transformation. *Electronic Notes in Theoretical Computer Science*, **152**, 161–173, 2006.
[23] SANS Institute. Csis: 20 critical security controls version 4.1, 2013.
[24] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi and D. Hughes. Using constraint programming to manage configurations in self-adaptive systems. *Computer*, **45**, 56–63, 2012.

[25] M. Schumacher. *Security Engineering with Patterns - Origins, Theoretical Models, and New Applications*, vol. 2754 of *Lecture Notes in Computer Science*. Springer, 2003.

[26] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley and Sons, Ltd, 2006.

[27] F. Semmak, C. Gnaho and R. Laleau. Extended kaos method to model variability in requirements. In L. A. Maciaszek, C. Gonzlez-Prez and S. Jablonski, (eds.) *Evaluation of Novel Approaches to Software Engineering*, vol. 69 of *Communications in Computer and Information Science*, pp. 193–205. Springer, 2010.

[28] S. A. Thomas. *SSL and TLS Essentials: Securing the Web with CD-ROM*. John Wiley & Sons, Inc, 2000.

[29] A. J. Varela-Vaca. OPBUS tools. http://www.lsi.us.es/ quivir/index.php/Main/AJVarelaOPBUS, 2012.

[30] A. J. Varela-Vaca. *OPBUS: A Framework for Improving the Dependability of Risk-aware Business processes*. PhD thesis, University of Seville, 2013.

[31] A. Jesus Varela-Vaca and R. M. Gasca. Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach. *Information and Software Technology*, **55**, 1948–1973, 2013.

[32] A. J. Varela-Vaca, R. Warschofsky, R. M. Gasca, S. Pozo and C. Meinel. A security pattern-driven approach toward the automation of risk treatment in business processes. In *International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions*, vol. 189 of *Advances in Intelligent Systems and Computing*, pp. 13–23. Springer, 2013.