Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach

Angel Jesus Varela-Vaca *, Rafael M. Gasca

Department of Computer Languages and Systems, ETS Ingeniería Informática, University of Seville, Avda. Reina Mercedes S/N, 41012 Sevilla, Spain[1]

A B S T R A C T

Keywords:

Business process

Business process management systems

Security

Risk treatments

Constraint programming

Feature model

Context: The use of Business Process Management Systems (BPMS) has emerged in the IT arena for the automation of business processes. In the majority of cases, the issue of security is overlooked by default in these systems, and hence the potential cost and consequences of the materialization of threats could produce catastrophic loss for organizations. Therefore, the early selection of security controls that miti-gate risks is a real and important necessity. Nevertheless, there exists an enormous range of IT security controls and their configuration is a human, manual, time-consuming and error-prone task. Furthermore, configurations are carried out separately from the organization perspective and involve many security stakeholders. This separation makes difficult to ensure the effectiveness of the configuration with regard to organizational requirements.
Objective: In this paper, we strive to provide security stakeholders with automated tools for the optimal selection of IT security configurations in accordance with a range of business process scenarios and orga-nizational multi-criteria.
Method: An approach based on feature model analysis and constraint programming techniques is pre-sented, which enable the automated analysis and selection of optimal security configurations.
Results: A catalogue of feature models is determined by analyzing typical IT security controls for BPMSs for the enforcement of the standard goals of security: integrity, confidentiality, availability, authorization, and authentication. These feature models have been implemented through constraint programs, and Con-straint Programming techniques based on optimized and non-optimized searches are used to automate the selection and generation of configurations. In order to compare the results of the determination of configuration a comparative analysis is given.
Conclusion: In this paper, we present innovative tools based on feature models, Constraint Programming and multi-objective techniques that enable the agile, adaptable and automatic selection and generation of security configurations in accordance with the needs of the organization.

1. Introduction

Current Business Process Management Systems (*BPMS*), such as *Intalio BPMS*, *AuroPortal*, *Bonita BPM*, *BizAgi*, and *jBPM*, are able to automatically generate business-process-driven software products without hardly any human intervention. In most cases, the security in BPMS is overlooked by default; furthermore, PricewarterhouseCoopers' report highlighted that the investment in security by organisations is no higher than one to three per cent of total budget [1]. For instance, *Bonita Soft* [2] is used to optimize the aircraft manufacturing process. The cost and consequences of the materialization of threats in these systems range from mildly annoying to catastrophic [3–5], with serious injury occurring or systems destroyed, reputation losses, security breaches, and so on. In general, *BPMS*, Information Systems (IS), communications, and business processes have applied security controls, such as anti-virus software, and periodical patches to systems. It is therefore crucial for organisations to act as soon as possible in selecting adequate security treatment according to the necessity of business objectives with respect to IT security risks. The automation of security controls will substantially lower the cost of security while improving its effectiveness [6].

The selection and configuration of security controls is one of the main problems within the scope of IT security since, in most cases it is a human, manual, time-consuming and error-prone task that involves several security stakeholders, such as security managers and administrators [7]. There exist an enormous number of controls [8] that can vary from abstract to specific controls depending on the level of the specification to be implemented (Fig. 1). For instance, one security control might involve the deployment of a

* Corresponding author.
E-mail addresses: ajvarela@us.es (A.J. Varela-Vaca), gasca@us.es (R.M. Gasca).
[1] http://www.lsi.us.es/~quivir.

procedure in order to carry out backups of crucial data. However, specific IT security controls can vary from the simple installation of anti-virus software to the configuration of secure protocols (i.e. HTTPS) or even the configuration of an Access Controls List (ACL) for a network firewall. This heterogeneity complicates the task of selection and configuration of specific security controls since it implies the involvement of a high level of knowledge and expertise. Furthermore, security managers and administrators are responsible for measuring whether the effectiveness of the security configuration implemented will comply with the organizational parameters of performance, cost, and level of acceptable risk. Ideally, this task should not lead to an increased workload for security managers and administrators.

In a previous work [9], we have treated the problem of using a representation of security controls. A formalization based on security patterns has been provided to specify security controls. This model enables the user to link security goals, descriptions of problems, and solutions. Furthermore, this formalization enables the inference of information through the model. Nevertheless, this work does not consider the inference of mechanisms and algorithms for the adequate and optimized selection of security controls.

In this paper, we propose providing security managers and administrators with new tools to place, collect and analyze the most suitable IT security controls for each business process scenario. It is assumed that an IT security control is defined as the statement of a configuration (value of a set of features) that enforces certain security goals. To this end, a feature analysis of certain IT security controls is first carried out to handle typical IT security risk referring to confidentiality, integrity, Authentication and Authorization (AA), and availability. Consequently, these feature analyses are employed to provide selection algorithms based on constraint programming techniques that aid in the selection of the best configuration, by taking IT security risks and business objectives into consideration.

This paper is structured as follows: Section 2 presents an overview of related work found in the literature; Section 3 introduces a formalization for the problem of selection of a risk treatment; Section 4 gives an introduction to feature-oriented model analysis concepts; Section 5 provides feature models for security controls for *BPMS*; Section 5.2 presents an analysis of feature models for the generation of security configurations; and in Section 6, conclusions are drawn and future work is proposed.

2. Related work

In previous work, the OPBUS (Optimization BUsiness process Security) framework is presented [10,11]. The framework has been developed as a technological solution to fill the gap between business process development, risk management and fault tolerance by means of the improvement of BPM life-cycle. OPBUS strives to provide business and security analysts with automatic tools for risk management at the design time of business processes. These tools provide mechanisms based on model-based diagnosis theory and constraint programming techniques that enable automatic risk assessment during the design of business process models [10,11]. OPBUS tools enable the user to carry out a risk evaluation of the business process and the identification of risk that exceeds the acceptable risk level established in the risk criteria. On the other hand, the framework also provides fault tolerance infrastructure in order to achieve dependable executable business processes at run-time [12]. The framework has also been supported by a range of developing tools [13]. The OPBUS tool enables business and security analysts to obtain an abstract risk assessment of business process models, by pointing out specifically where the business processes have to be treated. This assessment is based on an extension [10] of the business process properties, such as threats that affect each activity, the value of activities (in various security dimensions), and the frequency and consequence of threats. The business process models can be configured with risk information and transformed to constraint models in order to be evaluated. In Fig. 2, there is a screenshot which shows most of the features provided by the OPBUS tool. The tool provides an extended BPMN editor to support the extension proposed in [10]. The tool is equipped with a *Validation* option (cf. Fig. 2) that automatically transforms and carries out a risk evaluation of the model. In the *Console* tab, results of the risk evaluation can be observed. In the *Project Explorer* tab, a constraint model is generated within the project space. If any non-conformance is obtained the information is retrieved by) the editor with red signals and all non-conformance problems are collected in the *Problems tab*. The transformation of the model

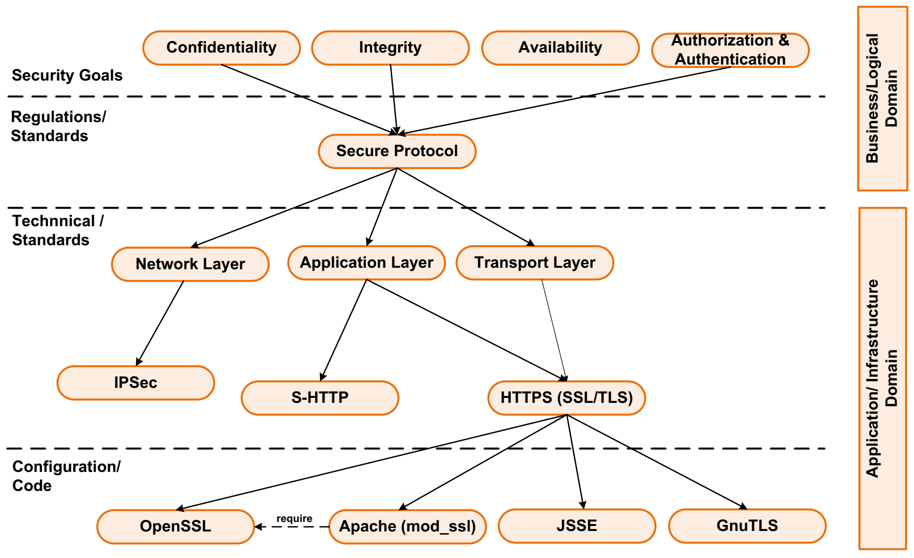
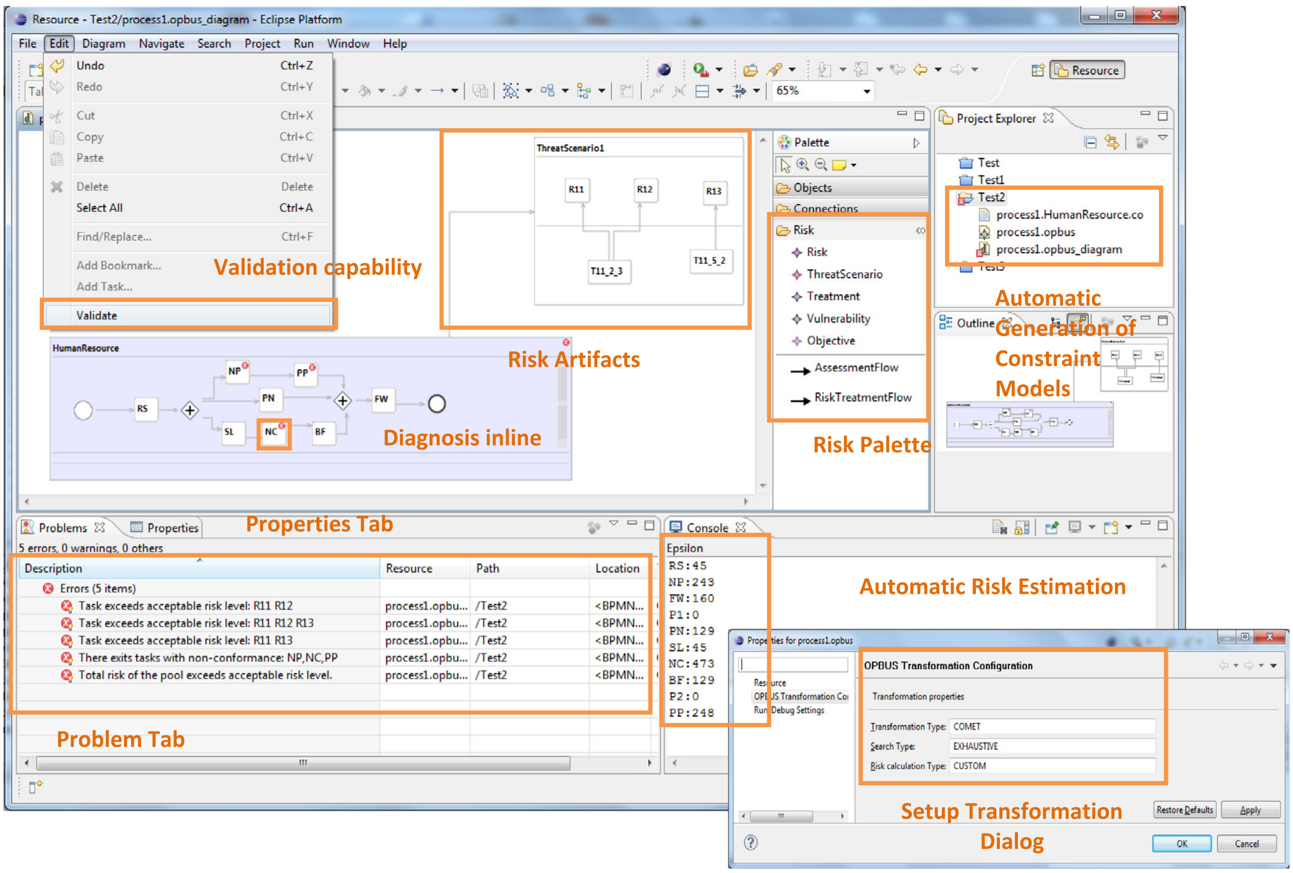


Fig. 1. Hierarchy of security controls.



5 errors, 0 warnings, 0 others

Description	Resource	Path	Location
Errors (5 items)			
Task exceeds acceptable risk level: R11 R12	process1.opbu...	/Test2	<BPMN...
Task exceeds acceptable risk level: R11 R12 R13	process1.opbu...	/Test2	<BPMN...
Task exceeds acceptable risk level: R11 R13	process1.opbu...	/Test2	<BPMN...
There exits tasks with non-conformance: NP,NC,PP	process1.opbu...	/Test2	<BPMN...
Total risk of the pool exceeds acceptable risk level.	process1.opbu...	/Test2	<BPMN...

```
Epsilon
RS:45
NP:243
FW:160
P1:0
PN:129
SL:45
NC:473
BF:129
P2:0
PP:248
```

OPBUS Transformation Configuration

Transformation properties

Transformation Type:	COMET
Search Type:	EXHAUSTIVE
Risk calculation Type:	CUSTOM

Fig. 2. Identification of business process activities with risk.

can be set up using different parameters such as shown Fig. 2. The figure shows a business process that has been assessed. As can be observed, the tool retrieves information about which threats are responsible for non-conformance within the editor over the activities; in this case *R11*, *R12 and R13* affect activities *NP*, *PP* and *NC*. Risks *R11* and *R12* might represent threats related to certain vulnerabilities; for instance *SQL injections* and *XSS-injections*. This information is highly useful in the identification of how and where the business process should be protected since a risk level, non-acceptable to the organization, is produced.

OPBUS lacks mechanisms for the inference, generation and selection (risk treatment) of adequate security controls that enable the user to modify the risk until risk levels are established as acceptable within business processes.

A number of initiatives have emerged in the context of BPM in order to bridge the gap between risk assessment and business domains [14–18]. Most of these initiatives are aimed at providing enhancements of business process languages through new domain-specific languages for the inclusion of assets, requirements, goals, and threats into business process models. These approaches are focused on extending models to enrich their expressivity in order to aid and improve the documentation of risk assessment in business processes. Nevertheless, none of these approaches provides a mechanism for the automation and identification of suitable treatment to mitigate risks.

There exist various initiatives related to the selection of security controls in business process models [19–22]. In [19], Rodriguez et al. propose an extension to UML 2.0 activity diagrams in order to provide graphical annotations for the specification of security requirements in the diagrams. This extension is defined by means of a UML profile called *BPSec*. Related to this work, in [21], Wolter et al. provide security annotations for graphical business processes that enable security configurations to be directly set up in the business process model. In [20], Menzel et al. define an extension of business process models that enable the generation of a security configuration of trust and data integrity with respect to a predetermined level of security risk for an SOA environment. Other more general approaches exist, such as that proposed in [23]. In [23], the authors provide a goal-driven approach as an extension of Tropos/i* in order to analyze risk at organization level. Furthermore, they illustrate a number of different techniques to help the analyst in identifying and enumerating relevant controls for risk mitigation.

There exist certain studies in the revised literature with respect to feature-oriented domain analysis and the generation of models [24–27]. Certain efforts in the literature revised are focused on the generation of requirement models. In [24], Semmak et al. propose an extension of a goal-driven method (KAOS) in order to generate adaptive requirements models from variant models. In [25], Mellado et al. propose an approach which facilitates the development of secure software product lines (SPLs) and their derived products. In [26], Pérez et al. use feature models to analyze the variability requirements and consequently transform this feature model 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 feature models. In [27], Sawyer et al. use feature model analysis 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 feature model, and by means of feature model analysis, 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 our approach, however, quantitative domains for attributes are supported.

3. A formal approach for security-risk treatment problems in a BPMS

As mentioned above, business processes are deployed and executed in BPM systems. In a *BPMS*, a set of n business process models $\{BP_1, BP_2, \ldots, BP_n\}$ could be executed for all $BP_i \in BPMS$, whereby there exists an *estimated risk* in the design time t such that $R(t)_{BPi} = \{R(t_1)_{BPi}, R(t_2)_{BPi}, \ldots, R(t_m)_{BPi}\}$, $1 \leqslant t \leqslant m$. For instance, the risk might be determined according to CRAMM (CCTA Risk Analysis and Management Method) methodology [28]: by the asset value (a value in the interval from 1 to 5), percentage of loss (a value in the interval from 1 to 5), and frequency (a value in the interval from 1 to 5). In this case, the risk estimation may be the sum of the three values. It must be borne in mind that OPBUS tools support uncertainty, i.e. a range of values for the specification of metrics, such as frequency $\in$ [2–5].

In order to estimate a risk, an assessment is performed which determines whether risks exceed the threshold imposed by the organization. In Fig. 3, the estimation of risks is depicted with regard to the five security dimensions and time. In the figure, it can be observed that the two first estimations exceed the risk threshold. However, after including risk treatment, the risk has decreased to the point where the optimum value is achieved. In the following definitions, it is assumed that risks are determined at a fixed point of time t_i. Therefore, at design time t_i for a *BPMS*, the associated risk is represented by $R(t_i)_{BPMS} = R_{BPMS}$.

There are two main problems related to the risk management of a *BPMS*: (1) the diagnosis of non-conformance risks within business processes of a *BPMS* regarding risk thresholds and (2) the treatment of these non-conformances through the selection of a set of controls. For a better understanding, both problems are formalized similarly to the formalization used by Cordier et al. [29].

In a *BPMS*, a set of n business process models $\{BP_1, BP_2, \ldots, BP_n\}$ could be executed. A business process model, BP_k, is composed of a set of $A^{BP_k} = \{A_1, A_2, \ldots, A_n\}$ activities that are attributed by $R_{BP_k} = \{R_{A_1}, R_{A_2}, \ldots, R_{A_m}\}$ IT security risks.

Definition 1. A risk of one activity R_{A_j} is represented by a 5-tuple defined as the combination of five security dimensions: integrity ($R^i_{A_j}$), confidentiality ($R^c_{A_j}$), availability ($R^a_{A_j}$) authorization ($R^{az}_{A_j}$) and authentication ($R^{at}_{A_j}$). In (1), vector notation represents the different security dimensions of R_{A_j}.

$$R_{A_j} = (R^i_{A_j}, R^c_{A_j}, R^a_{A_j}, R^{at}_{A_j}, R^{az}_{A_j}) \tag{1}$$

In the same way, the risk of a business process BP_k might be defined as the combination of R_{BP_k}, as proposed in [10]. In the listing (2), R_{BP_k} is represented by means of matrix notation. It is assumed here that there are no dependencies between risk and treatment.

$$R_{BP_k} = \begin{pmatrix} R_{A_j} \\ R_{Al} \\ \ldots \\ R_{A_m} \end{pmatrix} = \begin{pmatrix} R^i_{A_j} & R^a_{A_j} & R^c_{A_j} & R^{at}_{A_j} & R^{az}_{A_j} \\ R^i_{A_l} & R^c_{A_l} & R^a_{A_l} & R^{at}_{A_l} & R^{az}_{A_l} \\ & & \ldots & & \\ R^i_{A_m} & R^c_{A_m} & R^a_{A_m} & R^{at}_{A_m} & R^{az}_{A_m} \end{pmatrix} \tag{2}$$

Definition 2. The **risk of a *BPMS*** (R_{BPMS}), which contains a finite set of business processes $\{BP_1, BP_2, \ldots, BP_n\}$ is defined by the union of the risks of these business processes.

$$R_{BPMS} = \{R_{BP_1}, R_{BP_2}, \ldots, R_{BP_n}\} \tag{3}$$

Definition 3. A set of business goals (BG) establishes a set of constraints that relate the risks of a BPMS, and relate its business process models.

In general, organizations establish a set of business goals related to the appraised, i.e. a risk of a business process BP_k is acceptable if $R^i_{A_j} \leqslant n$, where n may be a natural number.

Definition 4. A **risk-management problem** is a tuple (R_{BPMS}, BG), where R_{BPMS} represents the system model to be diagnosed (for model-based diagnosis theory [30], components and observational model), and BG represents the business goals.

$$R_{BPMS} \cup BG \nvdash T \tag{4}$$

Thus, the system R_{BPMS} is unsatisfiable with regard to the BG constraints established. The diagnosis strives towards the identification of inconsistencies in the risk of an activity A_j, and specifically the set of dimensions (integrity, confidentiality, availability, authorization, and authentication) within the activities that produce that inconsistency. This identification might be solved by means of fault diagnosis theory. Following the theory of consistency-based diagnosis proposed by Kleer et al. in [30], this problem has been formalized as a *fault diagnosis of a risk-management problem* in the following definitions. This problem is solved in [11,13].

Definition 5. A **fault diagnosis of a risk management problem** is a set of risks $\Delta \subseteq R_{BP_k}$ such that $\Delta = \{\Delta_{BP_j} \cup \Delta_{BP_l} \cup \ldots \cup \Delta_{BP_m}\}$, where $\Delta_{BP_x} \subseteq R_{BP_x}$ and $x \in [j, \ldots, m]$.

$$(R_{BPMS} - \Delta) \cup BG \vdash T \tag{5}$$

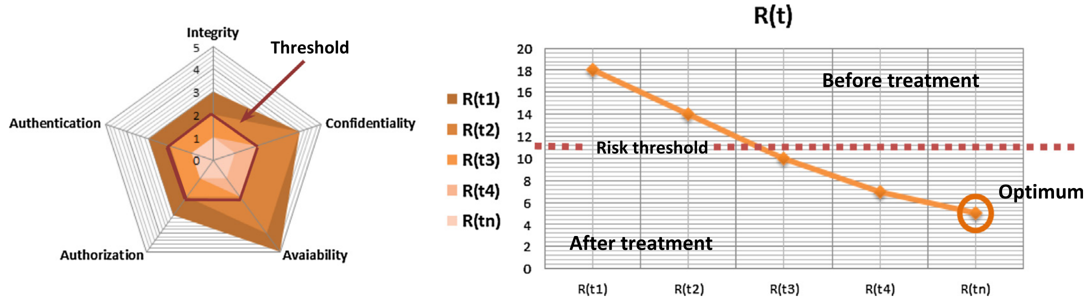


Fig. 3. Evolution of risk estimation with respect to time.

The fault diagnosis of a risk management problem identifies the set of activities of the business process BP_k, SC^c, which are responsible for the inconsistency. Since the activities are known, it is therefore possible to analyze which dimensions of security might be treated. In Fig. 4, an example of fault diagnosis of a BPMS is illustrated. The fault diagnosis indicates which activities and dimensions should be treated; for instance, the diagnosis of business process BP_2 is R_{A_1} and R_{A_2}, where R_{A_1} is only affected in the dimensions of confidentiality and availability, while R_{A_2} is affected in authentication and availability.

In (6), an example of $\varDelta_{BP_x}$ is represented by matrix notation, where the symbol $\sim$ has been used to indicate the dimensions omitted. For instance, the dimensions to be treated for R_{A_j} are integrity, and authentication, while the remaining dimensions are omitted.

$$\varDelta_{BP_k} = \begin{pmatrix} R_{A_j} \\ R_{Al} \\ \cdots \\ R_{A_m} \end{pmatrix} = \begin{pmatrix} R^i_{A_j} & \sim & \sim & R^{at}_{A_j} & \sim \\ \sim & R^c_{A_l} & \sim & R^{at}_{A_l} & \sim \\ & & \cdots & & \\ R^i_{A_m} & \sim & \sim & \sim & \sim \end{pmatrix} \tag{6}$$

At this point, the *BPMS* has been diagnosed by identifying the $\varDelta$ responsible for the inconsistency. The application of a risk treatment is proposed in order to mitigate these problems. In this paper, we strive to determine a set of controls SC that treats $\varDelta$. Hereinafter, it is assumed that, given a R_{BPMS} and $\varDelta$, the risk treatment problem could be formalized as follows:

Definition 6. A **risk-treatment problem** is defined by the tuple $(SC(x), \varDelta)$ where $SC(x)$ is the set of controls to be applied in x, and $\varDelta$ represents the activities to be treated. Therefore, the risk treatment problem becomes the identification of the set of controls, $SC(x)$, such that:

$$(R_{BPMS} - \varDelta) \cup SG(\varDelta) \cup BG \vdash T \tag{7}$$

Solving a risk treatment problem for a particular $\varDelta$, a set of controls is assumed, denoted as $SC(x)$.

Definition 7. A **control** $SC^d(x)$ is a security configuration that produces a risk reduction or mitigation in the dimension d of x, where $d \in \{i, c, a, at, az\}$.

For a better understanding, an illustrative example is given. Since business goals indicate that the risk in any dimension cannot exceed 15, let us consider $\varDelta$ as given in (8). It is assumed that the risks and controls are assigned to appropriate quantitative values. The estimation of these values relies on the expertise of various stakeholders and their experience to accurately estimate this data [62].

$$\varDelta = \begin{pmatrix} R_{A_j} \\ R_{A_h} \\ R_{A_k} \\ R_{A_l} \\ R_{A_m} \end{pmatrix} = \begin{pmatrix} R^i_{A_j} & \sim & \sim & \sim & \sim \\ R^i_{A_h} & R^c_{A_h} & \sim & \sim & \sim \\ \sim & \sim & R^a_{A_k} & \sim & \sim \\ \sim & \sim & \sim & R^{at}_{A_l} & \sim \\ \sim & \sim & \sim & \sim & R^{az}_{A_m} \end{pmatrix} = \begin{pmatrix} 15 & \sim & \sim & \sim & \sim \\ 17 & 16 & \sim & \sim & \sim \\ \sim & \sim & 16 & \sim & \sim \\ \sim & \sim & \sim & 18 & \sim \\ \sim & \sim & \sim & \sim & 19 \end{pmatrix} \tag{8}$$

For the diagnosis problem above, a set of security configurations $SC(x)$, is selected, such as that given in (9). The configuration $SC^i(x)$ modifies the dimension of the integrity risk of any activity. In this example, it can be observed that: in the first row, a 0.50 value that represents a 50% reduction is applied for $R^i_{A_j}$ in the dimension of integrity; in the second row, the 0.50 represents the percentage (50%) of risk reduction applied to integrity for $R^i_{A_h}$; 0.60 indicates the percentage (60%) of risk reduction applied to confidentiality for $R^c_{A_h}$ and so on.

$$SC(x) = \begin{pmatrix} 0.50 & \sim & \sim & \sim & \sim \\ 0.50 & 0.60 & \sim & \sim & \sim \\ \sim & \sim & 0.60 & \sim & \sim \\ \sim & \sim & \sim & 0.60 & \sim \\ \sim & \sim & \sim & \sim & 0.50 \end{pmatrix} \tag{9}$$

In order to illustrate the calculation of $SC(\varDelta)$ the operator $\otimes$ has been defined to represent a matrix, such that for $R = A \otimes B$, the entry r_{ij} is the result of multiplying the entry a_{ij} of A by b_{ij} of B. For instance, in (10), the entry 7.5 is the result of multiplying the $(1,1)$th entry of $\varDelta$ by the $(1,1)$th entry of $SC(x)$. The final result is a matrix that indicates the new values for $\varDelta$ In this case, the risk values comply with the limitations set out in the business goals which state that risk values must not exceed 15.

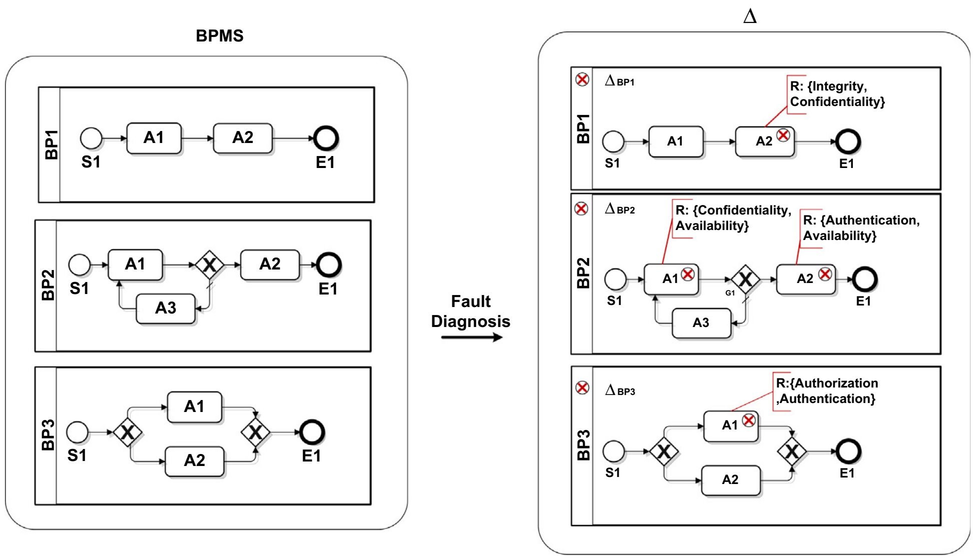

BPMS
BP1
S1
A1
A2
E1
BP2
A3
BP3
Fault Diagnosis
Δ
Δ BP1
R: {Integrity, Confidentiality}
Δ BP2
R: {Confidentiality, Availability}
R: {Authentication, Availability}
G1
Δ BP3
R:{Authorization ,Authentication}

Fig. 4. Fault diagnosis of a BPMS.

$$SC(\Delta) = \begin{pmatrix} 0.50 & \sim & \sim & \sim & \sim \\ 0.50 & 0.60 & \sim & \sim & \sim \\ \sim & \sim & 0.60 & \sim & \sim \\ \sim & \sim & \sim & 0.60 & \sim \\ \sim & \sim & \sim & \sim & 0.50 \end{pmatrix} \otimes \begin{pmatrix} 15 & \sim & \sim & \sim & \sim \\ 17 & 16 & \sim & \sim & \sim \\ \sim & \sim & 16 & \sim & \sim \\ \sim & \sim & \sim & 18 & \sim \\ \sim & \sim & \sim & \sim & 19 \end{pmatrix} = \begin{pmatrix} 7.5 & \sim & \sim & \sim & \sim \\ 8.5 & 9.6 & \sim & \sim & \sim \\ \sim & \sim & 9.6 & \sim & \sim \\ \sim & \sim & \sim & 10.8 & \sim \\ \sim & \sim & \sim & \sim & 9.5 \end{pmatrix} \tag{10}$$

In Fig. 5, the result of applying controls $SC(x)$ to Δ is illustrated. The result, $SC(\Delta)$, is the same *BPMS* whose activities have a set of controls that mitigate the risk in the dimension affected. For instance, activity *A2* within BP_1 has two controls, one for the integrity (*SCi*), and the other for the confidentiality (*SCc*).

On the other hand, the suitability of the configurations with respect to the requirements defined by the organization must be measured. In general, organizations might be interested to measure, for instance, cost-benefit, profit, or Return Of Security Investment (ROSI). To this end, metrics could be related to features of controls in order to measure costs, delays, etc. Definition 5 can therefore be reformulated as follows:

Definition 8. A **risk-treatment problem** is defined by the tuple ($SC(x)$, M, Δ), where $SC(x)$ is the set of controls to be applied; Δ represents the activities to be treated; and M are the metrics to be considered related to controls.

In (12), there is an example of costs related to the controls. For instance, the entry 100 indicates the cost in monetary units of the configuration selected for $R^i_{A_j}$.

$$M_{\cos t} = \begin{pmatrix} 100 & \sim & \sim & \sim & \sim \\ 250 & 350 & \sim & \sim & \sim \\ \sim & \sim & 500 & \sim & \sim \\ \sim & \sim & \sim & 1000 & \sim \\ \sim & \sim & \sim & \sim & 600 \end{pmatrix} \tag{11}$$

In most cases, it could prove interesting to find the best configuration according to certain objective functions. In this case, an optimization function is required in order to determine the best solution with regard to the metrics M (i.e. cost, time of delay, etc.). Hence, the problem is reformulated yet again to determine risk treatment problems that optimize certain objective functions.

Definition 9. An **optimal risk-treatment problem with a single objective** is defined by the tuple $(F, SC(x), M, \Delta)$ where $SC(x)$ is the set of controls to be applied; M is the set of metrics to be considered; F is the objective function (in the literature denoted by MAX or MIN); and Δ represents the activities to be treated, such that:

$$\begin{aligned} & F \equiv \text{MIN}(SC(x), M) \text{ or } \text{MAX}(SC(x), M) \equiv SC'(x) \\ & \text{subject to,} \\ & (R_{BPMS} - \Delta) \cup SC'(\Delta) \cup BG \vdash T \end{aligned} \tag{12}$$

Thus, the problems consist of the identification of configurations that optimize the objective function. In the following listings, an example of the selection of configurations that minimizes the cost is presented. In this example, there are two sets of configurations $SC^1(x)$ and $SC^2(x)$, whose costs are $M^{SC^1}_{\cos t}$, and $M^{SC^2}_{\cos t}$, respectively. The application of the objective function results in a new set of controls $SC'(x)$ adjusted to the minimum cost. For instance, for the $(1,1)$th entry for $SC^1(x)$ and $SC^2(x)$, the best control is that given in $SC^1(x)$ since it presents a better risk reduction, although it is more expensive than $SC^2(x)$. In this case, the configurations of $SC^2(x)$ have been selected since they minimize the cost (objective function).

$$SC^1(x) = \begin{pmatrix} 0.50 & \sim & \sim & \sim & \sim \\ 0.50 & 0.60 & \sim & \sim & \sim \\ \sim & \sim & 0.60 & \sim & \sim \\ \sim & \sim & \sim & 0.60 & \sim \\ \sim & \sim & \sim & \sim & 0.50 \end{pmatrix}, \quad \text{and } SC^2(x) = \begin{pmatrix} 0.50 & \sim & \sim & \sim & \sim \\ 0.50 & 0.70 & \sim & \sim & \sim \\ \sim & \sim & 0.80 & \sim & \sim \\ \sim & \sim & \sim & 0.30 & \sim \\ \sim & \sim & \sim & \sim & 0.60 \end{pmatrix} \tag{13}$$

$$M^{sc^1}_{\cos t} = \begin{pmatrix} 300 & \sim & \sim & \sim & \sim \\ 250 & 350 & \sim & \sim & \sim \\ \sim & \sim & 500 & \sim & \sim \\ \sim & \sim & \sim & 1000 & \sim \\ \sim & \sim & \sim & \sim & 600 \end{pmatrix}, \quad \text{and } M^{SC^2}_{\cos t} = \begin{pmatrix} 200 & \sim & \sim & \sim & \sim \\ 250 & 450 & \sim & \sim & \sim \\ \sim & \sim & 600 & \sim & \sim \\ \sim & \sim & \sim & 800 & \sim \\ \sim & \sim & \sim & \sim & 650 \end{pmatrix} \tag{14}$$

$$\text{Min}(\{SC^1(x), SC^2(x)\}, \{M^{SC^1}_{\cos t}, M^{SC^2}_{\cos t}\}) = SC'(x) = \begin{pmatrix} 0.50 & \sim & \sim & \sim & \sim \\ 0.50 & 0.60 & \sim & \sim & \sim \\ \sim & \sim & 0.60 & \sim & \sim \\ \sim & \sim & \sim & 0.60 & \sim \\ \sim & \sim & \sim & \sim & 0.50 \end{pmatrix} \tag{15}$$

In particular cases, it is crucial to find the best configurations according to multiple objectives, such as cost, functionality, and performance. The problem lies in the optimization of searching within a set of controls that complies with G objectives. The problem is reformulated to determine the set of controls that optimize multiple-objective functions.

Definition 10. An **optimal risk-treatment problem with multiple objectives** is defined by the tuple $(F_G, SC(x), M, \Delta)$ where $SC(x)$ is the set of controls; M are the set of metrics to be considered; F_G is a set of optimization functions $\{F_{g_1}, F_{g_2}, \ldots, F_{g_n}\}$, (in the literature indicated by MAX or MIN); and Δ represents the activities to be treated.

$$\begin{aligned} & F \equiv \text{MIN}(SC(x), M) \text{ or } \text{MAX}(SC(x), M) \equiv SC'(x) \text{ such that,} \\ & F_G = \bigcup_i F_{g_i} \\ & \text{subject to,} \\ & (R_{BPMS} - \Delta) \cup SC'(\Delta) \cup BG \vdash T \end{aligned} \tag{16}$$

As mentioned earlier, the main problem of risk-treatment is the selection of the controls, $SC(x)$, to be applied in those activities with a problem of risks, Δ, together with the constraints of the problem R_{BPMS}, make the problem satisfiable. The selection of a set of controls that are suitable for a set of risks is a NP-hard problem similar to that presented in [31]. The aim of this paper is to provide mechanisms for the automatic selection and generation of security configurations for *BPMS* in order to force business processes to comply with business goals which rely on security risks.

Numerous controls exist, each with hundreds of possible characteristics and configurations that might be implemented [8].

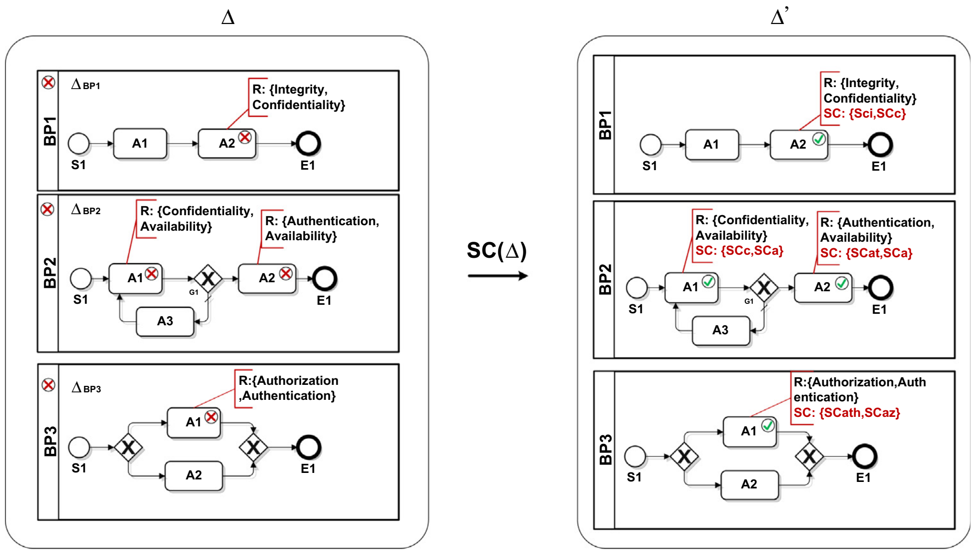

Δ
Δ'
SC(Δ)
BP1
BP2
BP3
R: {Integrity, Confidentiality}
R: {Confidentiality, Availability}
R: {Authentication, Availability}
R:{Authorization ,Authentication}
SC: {Sci,SCc}
SC: {SCc,SCa}
SC: {SCat,SCa}
R:{Authorization,Authentication}
SC: {SCath,SCaz}
S1
A1
A2
A3
E1
G1

Fig. 5. Application of controls to Δ.

We propose carrying out an analysis of the features of the main controls (IT security controls) at the infrastructure level of *BPMS*. This study is crucial for the definition of a catalogue of feature models with core features that might be used in the dimensions: confidentiality (SC^c), authorization (SC^{az}), authentication (SC^{at}), integrity (SC^i), and availability (SC^a). We also propose using feature model analysis methods in order to generate configurations in a flexible, agile, and automatic way. To this end, in the following section, concepts and formalisms of feature-oriented domain analysis are introduced.

4. Feature-Oriented Domain Analysis (FODA) for IT security

Software Product Line (hereinafter SPL) [32–34] 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 into software product line are focused on the domain analysis that consists of the process of analyzing related products in order to identify their common and variable features. Feature models (hereinafter FM) [33] is the most popular method for the domain analysis of a software product line. Feature models 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 [33], although the most widely used is that proposed by Czarnecki [34] as shown in Fig. 6. This representation enables four relations between a parent feature and its child features:

- *Mandatory*, child feature is required (*A* mandatory sub-feature of *B*, $A \leftrightarrow B$).
- *Optional*, child feature is optional (*A* optional sub-feature of *B*, $A \rightarrow B$).
- *Alternative*, one of the sub-features must be selected ($a_1 \ldots a_n$ alternative sub-feature of $b, a_1 \wedge \ldots \wedge a_n \leftrightarrow b \wedge_{i<j}(a_i \vee a_j)$).
- *Or-relation*, at least one of the sub-features must be selected ($a_1 \ldots a_n$ or sub-feature of b, $a_1 \vee \ldots \vee a_n \leftrightarrow b$).

In addition, other relations (cross-tree constraints) are allowed. The most common are:

- Feature *A requires* feature *B* ($A \rightarrow B$).
- Feature *A excludes* feature *B*($\neg(A \wedge B)$).

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 [35,37] 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). Fig. 6 shows an example of a feature model extended with attributes for the *Protocol* feature.

Several techniques are available for the automated analysis of FMs [34], using Propositional Logic (PL), Constraint Programming (CP), and Description Logic (DL). These approaches transform the feature models 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).

Constraint Programming, similar to that proposed in [35], is employed to carry out feature model analysis since this approach enables integer domains to be used for attributes and optimization functions. In [35], the authors apply a transformation to Constraint Satisfaction Optimization Problem (COP) of extended feature models which is able to automatically obtain information about the model. Fig. 7 gives an example of transformation of a feature model to a COP whose objective function is to find the maximum of *x*.

As aforementioned, we propose determining and inferring which configuration is the best possible according to the criterion (optimization function) by means of automated feature model analysis. To this end, it is necessary: (1) to define a catalogue of feature models and (2) to provide mechanisms for automatic selection and generation of configurations. In the following section, a catalogue of feature models is presented which is related to various IT security controls for *BPMS* in order to enforce security goals

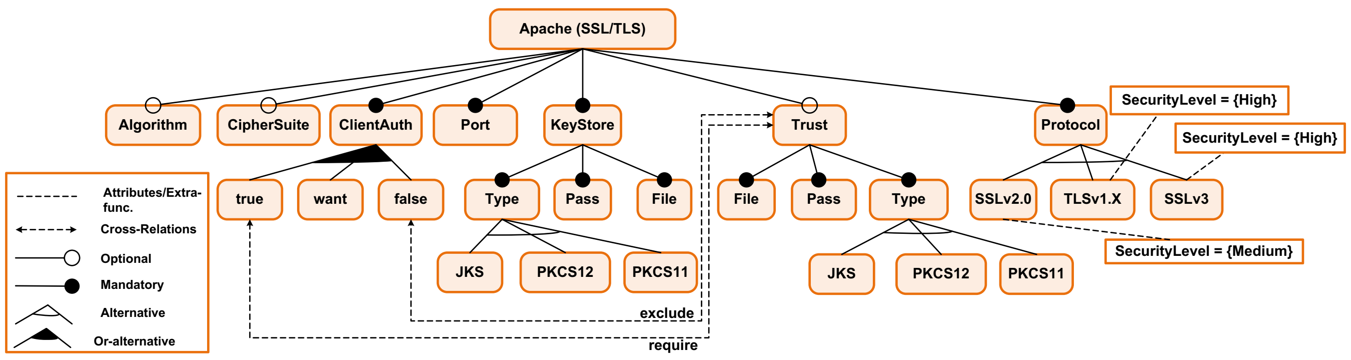


Fig. 6. Example of feature model for SSL/TLS configuration in Apache Tomcat.

within business processes. In the subsequent section, a case study is presented of the results for the selection and generation of optimal configurations using Constraint Programming techniques. Various optimization techniques have been applied in order to infer configurations that respond to the organizational requirements.

5. Case study: IT security controls in a real BPMS

The aim of the case study is to show a real scenario where automatic selection and generation of security configurations (IT security controls according to Definition 7) can be applied and how these configurations can be suited to one or several objectives for the organization.

The case study presents an infrastructure such as shown in Fig. 8. This scenario and configuration has been used within research projects that have been successfully certified by the standard ISO/IEC 27001 Information technology – Security techniques – Information security management systems – Requirements [36]. Business analysts define business process models as mere formalizations, and hence consider them as blueprints. Subsequently, those business process models are given to IT stakeholders for their implementation, thereby rendering IT stakeholders responsible not only for the decision of developing business processes in a specific technology, but also for how these business processes are used, and interconnect and communicate with other systems. As mentioned earlier, business processes are deployed and executed in *BPMSes* that communicate with other systems (Web Services, databases, web application servers, etc.). This communication creates a channel of communications between the *BPMSes* where business processes and other systems are executing as shown in Fig. 8. The example shows the communication between customers and business processes. In most cases, channels must be protected against possible IT security risks (threats or vulnerabilities) since this channel could transport crucial and confidential information.

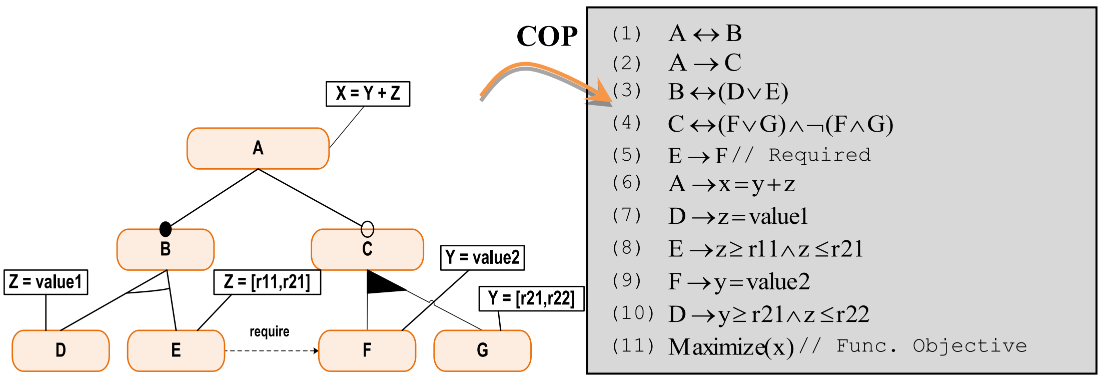
In Fig. 8, the infrastructure is made up of three main components or participants: (1) *Customers* that use business processes transparently through a proxy and BPMS. (2) A *Transparent proxy* used as a filter for requests from the customer side and responses from internal requests. In this case, we have used a *Load Balancer* that redirects external requests to a *Web Application Firewall* (*WAF*). The *WAF* inspects the request against a rule set to seek block malicious codes. Once the request is accepted by the *WAF* the request is delivered to the corresponding *BPMS*. (3) *BPMS* dispatch requests and responses from-to business processes. In the case a business process needs to use an external resource, such as an external service, *BPMS* must request a petition from the *Load Balancer* and the *Load Balancer* from Internet and vice versa.

Most leading *BPMSes* today use an infrastructure based on web technology in the form of web application servers with support to web service technologies. For instance, *Bonita Soft* can be used over *Apache Tomcat* or *JBoss* server infrastructure as a deploy server; *Intalio BPMS* use an *Apache ODE BPEL* engine; *BizAgi* supports a variety of servers such as *JBoss*, *Oracle WebLogic* and *IBM WebSphere*. *Bonita Soft* under *JBoss server* and *ModSecurity* under *Apache* with *Logs* are used as *BPMS* and WAF in the case study. These systems support the configurations and the integration of security controls in many different ways.

In the next subsections, we analyze configurations of *WAF*, *BPMS*, and business processes in order to achieve confidentiality, integrity, availability, authorization, and authentication. Furthermore, we provide techniques in order to infer configurations with regard to different multi objective functions.

5.1. Catalogue of feature models

According to Definition 7, the following subsections present a detailed description of the feature analysis carried out to achieve



(1) $A \leftrightarrow B$
(2) $A \rightarrow C$
(3) $B \leftrightarrow (D \vee E)$
(4) $C \leftrightarrow (F \vee G) \wedge \neg(F \wedge G)$
(5) $E \rightarrow F$ // Required
(6) $A \rightarrow x = y + z$
(7) $D \rightarrow z = value1$
(8) $E \rightarrow z \geq r11 \wedge z \leq r21$
(9) $F \rightarrow y = value2$
(10) $D \rightarrow y \geq r21 \wedge z \leq r22$
(11) Maximize(x) // Func. Objective

Fig. 7. Example of transformation of a feature model to a COP.

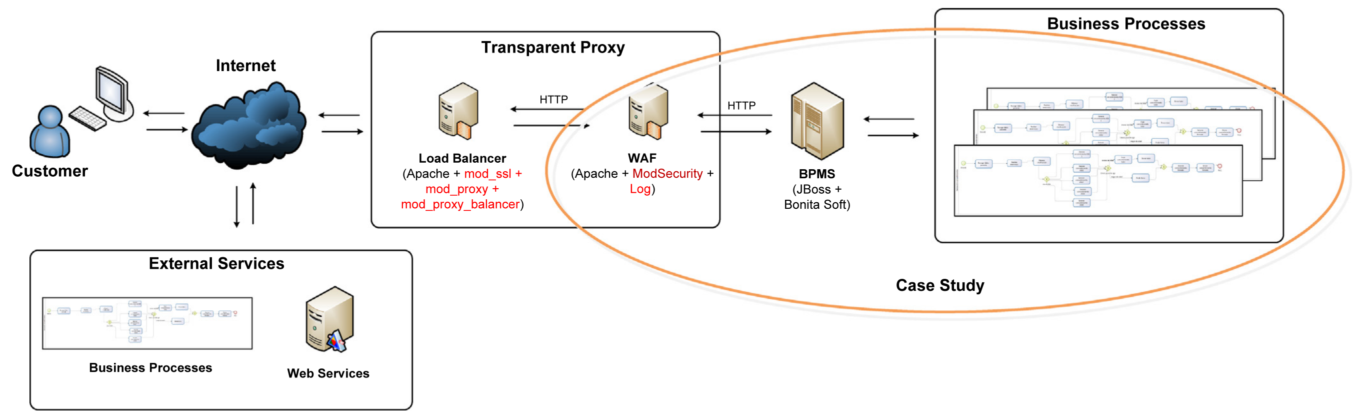


Fig. 8. Scenario of BPMS to be enforced by security controls.

configurations with regard to confidentiality (SC^c), authorization (SC^{az}), authentication (SC^{at}), integrity (SC^i), and availability (SC^a) in BPMS.

5.1.1. Feature model for availability

In accordance with the dependability taxonomy provided by Avižienis [38], a business process fails when the delivered service deviates from correct service. Fault tolerance strives to preserve the delivery of correct service in the presence of active faults. Fault tolerance is generally implemented by error detection and subsequent system recovery. The improvement in availability of business processes thanks to fault tolerance techniques is studied in [39,40,12]. In [12], various fault tolerance patterns are presented that may be integrated into *BPMS*. These patterns provide mechanisms for error detection and recovery (see Fig. 9). Furthermore, these techniques present an advantage since they might be integrated independently of the *BPMS* used.

Proposed fault tolerance patterns vary greatly, and include dynamic binding of services (Binder and Binder Backup), N-Version Programming (NVP) components, and the check-pointing approach. Each solution requires the configuration of the necessary components. For instance, the dynamic binding solution requires the specification of the *number of replicas* for each service; the specification of an *oracle*, which determines whether the solutions obtained are correct; and a *binder* component, which allows the dynamic binding of services at run-time. Nevertheless, NVP solutions require the selection of a *number of variants* for services and a kind of *adjudicator*. The NVP technique does not require an *oracle*. The complete description of the features and characteristics of these three fault tolerance patterns are given in Table 1.

The selection of one or another pattern presents a casuistry that has been modeled into the feature model (SC^a) as shown in Fig. 10. In this case, the feature model is based on those components necessary for the configuration of one of the fault tolerance proposals. Fault tolerance can be set up in business processes by selecting one of three possible options: (1) Dynamic Binding; (2) NVP; and (3) Checkpointing. For instance, a Checkpointing configuration needs at the first level the mandatory components of: Number of Sensors, Oracle (cf. oracle is a testing concept often used in test-driven paradigm. It represents as the intuition or wisdom of an human expert/component about of the expected result of a process for a given input), and Compensation Handlers. However, Oracle components require as mandatory a Diagnoser and a Business Rule engine. Thus, a valid Checkpointing configuration would be {Checkpointing, Number of sensors, Oracle, Compensation handlers, Diagnoser, Business Rules Engine}. Therefore, this feature model can help how to select and to configure a security control with respect to the availability in accordance to the fault tolerance patterns in [12].

Feature models can be extended with attributes and extra functionalities such as shown in Section 4. For this case study, we have decided only to extend the feature model for availability and confidentiality. The feature model for fault tolerance has been extended with attributes and extra functionalities. The details of the extension are given in Appendix D. In Table 2, there is a summary of attributes used to extend the feature model.

One of the main challenges in the field of security is the measurement of security [66]. In general, the units and values employed to determine a risk assessment process are based on statistical data, real-world reports, expertise of users, administrators, and the tracking of assets [62]. In all these cases, these values and units are related to real units. However, there exist cases whereby it is necessary to define abstract scales. In Table 5, the units, range, and description of attributes are defined. It must be borne in mind that different scales are applied, depending on the attribute. Certain attributes relate to a real unit (e.g. *Time*), while for others a generic range of values is used (e.g. *Number of Services*).

In fault tolerance, a very interesting attribute for the selection could be the *Mean Time To Repair* (*MTTR*). This attribute is defined as the time required to repair a system failure. In our case, by ignoring the time of detection, the time to repair might be defined as the sum of the delays introduced by each feature. For instance, the *MTTR* of a dynamic binding solution is the sum of the time of execution of *n* replicas (*NoR*), the execution of the *oracle*, and the time of execution of the *binders*. Attributes have been included to enable the calculation of the *MTTR* for each fault tolerance solution.

$$\begin{aligned} MTTR &= NoR \times NoR.\mathrm{Time} + \mathrm{Oracle.Time} + \mathrm{Binder.Time} \\ &\quad + \mathrm{BinderBackup.Time} \end{aligned} \tag{17}$$

The values for *MTTR* have been obtained from a previous work [12] where a performance study of fault tolerance patterns was carried out. In listing (18), there is an example that states NoV delay time is between 10 and 30 ms. Thus, delay time used by variants in a NVP configuration varies from 10 to 30 ms. On the other hand, the time of delay used by services in a configuration of checkpointing (CP) varies from 10 to 40 ms. Thus, NVP's MTTR could be less than CP's MTTR depending on the number of variants and services used for each solution.

$$NVP \rightarrow NVP.NoV\_\mathrm{Time} \geq 10 \text{ and } NVP.NoV\_\mathrm{Time} \leq 30 \tag{18}$$

$$\mathrm{CP} \rightarrow \mathrm{CP.NoS\_Time} \geq 10 \text{ and } \mathrm{CP.NoS\_Time} \leq 40 \tag{19}$$

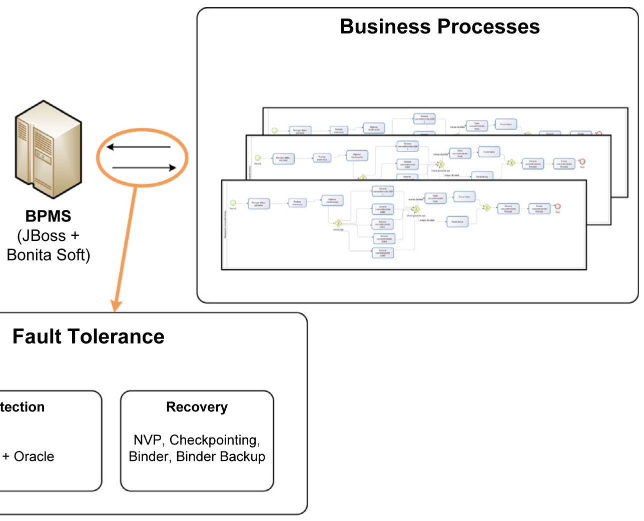


Fig. 9. Scenario for availability and reliability enforcement in BPMS.

Table 1
Description of features for fault tolerance patterns.

Feature	Dynamic Binding (DB)	N-Version Programming (NVP)	Check Pointing (CP)	Description
Number of Services (NoS)	√			*Indicates the number of Web Services used as backup*
Binder	√			*Indicates the utilization of a binder component*
Backup Binder (BBinder)	√			*Indicates the utilization of a backup for the binder*
Oracle	√		√	*Indicates whether an oracle is used. Oracle is composed of a Diagnoser and a Business Rule engine*
Number of Variants (NoV)		√		*Indicates the number of variants used within NV-Components*
Adjudicator		√		*Indicates the kind of adjudicator systems used. Four alternatives are proposed: Exact majority, Median, Mean, and Consensus*
Number of sensors (NoS)			√	*Indicates the number of check points used*
Compensation handlers			√	*Indicates the number of compensation handlers used*

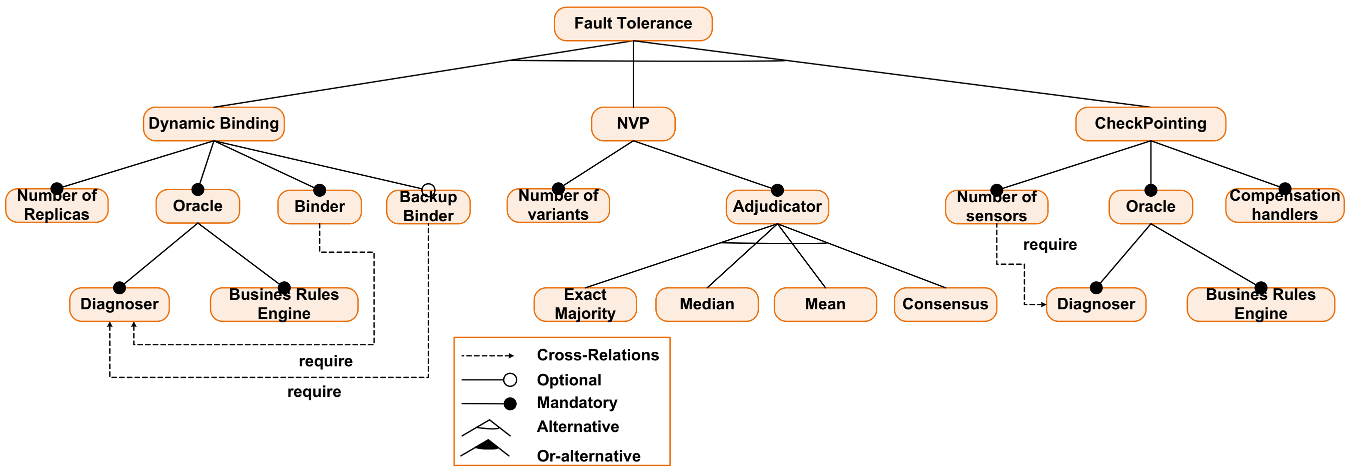


Fig. 10. Feature model for fault tolerance.

Another attribute included in the extension is risk reduction. As mentioned earlier, OPBUS enables the determination of which risks within the business processes must be treated [10,11]. OPBUS supports multiple risk formulas for risk assessment. Using the risk formula given in [10], where $Value_{Asset}$ is the combination (sum) of the five metrics (*integrity*, *confidentiality*, *availability*, *authorization*, *and authentication*) employed to obtain the asset value, and where *Consequence* and *Frequency* refer to properties of a particular threat. In the case of controls, these include adjustments by means of risk reduction that subtracts value from *Consequence* and/or *Frequency* of threats. This risk formula and its metrics are very similar to the MAGERIT method proposal [67], which defines the risk estimation by means of the multiplication of the value of one asset, of impact, and of frequency of one threat.

Table 2
Description of attributes to extend fault tolerance feature model.

Attributes	Units	Range	Description
Time	Milliseconds	Integer	Indicates the time of delay introduced for the feature
MTTR	Milliseconds	Integer	Indicates the mean time to repair a system failure
Number of services	*N° services*	[0–50]	Indicates the number of services/variants/sensors used (NoS_N, NoV_N, NoS_N)
Risk reduction	%	[0–100]	Indicates the percentage of reduction of risk (RR)

Nevertheless, the main problem is how to measure the effectiveness of controls, for instance, to ascertain the risk reduction achieved when using a particular configuration of one control. In certification processes (such as *ISO/IEC 27001*), this information is first estimated by following a certain range of values and boundaries agreed by the work team and the clients. Subsequently, assessments are adjusted by the information obtained in the tracking of assets and controls.

$$R_{BP_i} = (\text{Value}_{\text{Asset}}) \times (\text{Consequence}_{\text{Threat}} - \text{Risk}_{\text{Reduction}_c}) \times (\text{Frenquency}_{\text{Threat}} - \text{Risk}_{\text{Reduction}_F}) \quad (20)$$

Using the previous risk formula, the only way to reduce the risk is to include controls that reduce factors of consequence ($\text{Risk}_{\text{Reduction}_c}$) or frequency ($\text{Risk}_{\text{Reduction}_F}$). As mentioned earlier, the main aim of selecting configurations is to search for configurations that produce a sufficiently large risk reduction to render the risks under the established limit levels. However, it is very complex to measure (in a complete manner) how much risk reduction is produced by a specific control against certain threats [66]. Our proposal includes the introduction of attributes that aid the estimation of the percentage of risk reduction reached with regard to the *Consequence* and the *Frequency* of threats. This information could first be obtained based on the expertise, and subsequently this information must be improved in the revision process in order to obtain better configurations.

In the case of risk reduction, it can be observed in Appendix D that the risk reduction ranges are stated by taking into consideration the features and the values of attributes, such as the number of variants (*NoV_N*) used in an *NVP* solutions. In listing (21), there is a statement that relates the risk reduction with the number of variants used in a NVP configuration. In that case, if the number of variants is between 3 and 6 then the risk reduction achieved is equal to, or greater than, 60%.

$$NVP.NoV \rightarrow (NVP.NoV\_N \geq 3 \text{ and } NVP.NoV\_N \leq 6) \rightarrow NVP.\text{RiskReduction} \geq 60\% \quad (21)$$

The main aim in extending the feature models with attributes and functions is to provide multi-criteria for the selection of configurations such as stated in Definitions 9 and 10.

5.1.2. Feature model of confidentiality, integrity and authentication

As mentioned previously, there exist channels of communication between business processes with other systems. To ensure the integrity and confidentiality of information exchanged in these communications, a secure channel is mandatory (Fig. 11). A secure channel requires the application of digital signatures and encryption infrastructure. In Fig. 11, a secure channel is linked with rounded-corner rectangles that describe security intentions (left-hand side), and the characteristics (right-hand side) necessary for their attainment. This infrastructure can be applied at the transport (SSL/TLS) or message (WS-Security) layer. Secure Socket Layer (SSL) protocol [41] and Transport Layer Security (TLS) protocol [42], (hereinafter SSL/TLS) are widely used to provide confidentiality, authentication, and integrity in data communications. SSL/TLS provides three main security services: confidentiality, by encrypting data; message integrity, by using a message authentication code (MAC); and authentication, through digital signatures. SSL/TLS allows the authentication of both parties, server authentication with an unauthenticated client, and total anonymity. A detailed analysis of these protocols is given in Appendix A.

Most application servers for common application webs 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. In general, these servers designate connectors that may be set up to use certain providers. There are two main providers: widely used *JSSE* (Java Security Socket Extension) [43], and *OpenSSL* [44]. However, other providers exist, such as *GnuTLS* [45]. These providers provide good support for the SSL/TLS standard, and further features are available. For instance, *GnuTLS* supports *OpenPGP* certificate infrastructure. Application servers enable connectors to be established that use a specific suite of protocols and algorithms. Fig. 12 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. A summary of the features considered is given in Table 3. Following the standards, connections negotiate a CipherSuite (see details in Appendix A) to be used in the communication. CipherSuite presents an enormous combination of configuration since each Key Change Method can be combined with a number of Cipher and MAC algorithms.

In Appendix A, a detailed analysis for the current version of SSL/TLS standards is given. Fig. 13 shows the resultant feature models (SC^i, SC^{at}, and SC^c) of the analysis carried out for 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. The feature model only shows the values that are supported for each feature, although numerous constraints interrelate these features. Due to the lack of expressivity of feature models, 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 Appendix B, complete cross-tree constraints related to the FM of SSL/TLS are given.

Finally, it should be borne in mind that it is very hard to separate the configuration of controls for only data integrity or data confidentiality since confidentiality implies the application of integrity and authentication mechanisms. In this section, a feature model for the configuration of three security dimensions is defined. In the analysis of configurations, however, the independence of controls is assumed. Thus, configurations for confidentiality, integrity, and authentication could be selected separately.

Similar to availability, feature model of SSL/TLS has been extended with attributes and extra functionalities. The details of the extension are given in Appendix D. In Table 4, there is a summary of attributes used to extend the feature model. In this case, we have selected a set of attributes different to those selected for

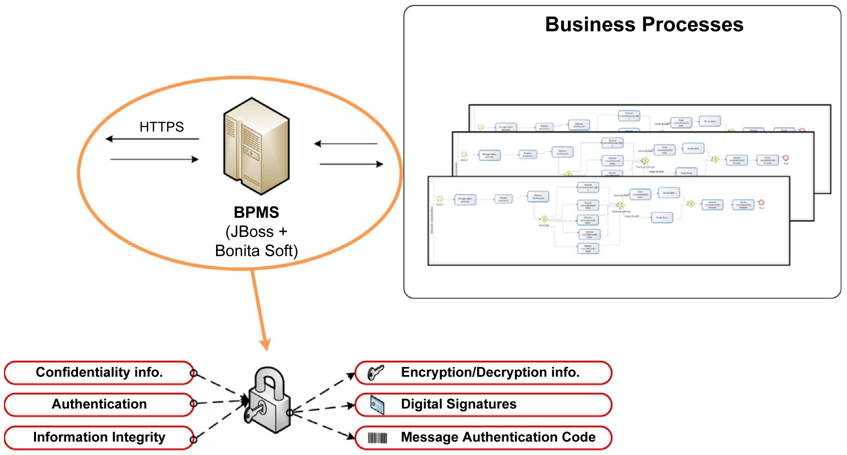


Fig. 11. Scenario for confidentiality, integrity, and authentication enforcement in BPMS.

```
<Call name="addSSLSocketConnector">
  <Arg>
    <New class="org.mortbay.jetty.security.SslSocketConnector">
      <Set name="Port">8443</Set>
      <Set name="maxIdleTime">60000</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>
          <Item>TLS_RSA_WITH_AES_256_CBC_SHA</Item>
          ….
        </Array>
      </Set>
    </New>
  </Arg>
</Call>
```

Fig. 12. Example of SSL/TLS configuration for a Jettyserver.

availability that enable the user to measure the cost-benefit of the configuration.

Regarding *SSL*/*TLS*, various metrics, typically used to measure the cost-benefit of security, have been included, as shown in Appendix D. Currently, the most important information for an organization is Return on Investment for any security investment (*ROSI*). In general, *ROSI* is calculated using the following formula [14]:

$$ROSI = \frac{\text{Risk Exposure} \times \text{Risk Migitation}\ (\%) - \text{Cost}}{\text{Cost}} \tag{22}$$

Following the proposal [14], in order to estimate the risk of exposure, the Annual Loss Exposure (*ALE*) can be used. *ALE* is calculated by multiplying the projected cost of a security incident (Single Loss Exposure, or *SLE*) with its estimated annual rate of occurrence (*ARO*). There are no standard methods that describe how to measure *SLE* and *ARO*. As mentioned earlier, this information is estimated based on statistical data, expertise, and the tracking of assets and controls.

$$ALE = SLE \times ARO \tag{23}$$

Assuming that *SLE* is fixed, the only way to reduce *ALE* is by reducing the *ARO*. The purpose for the introduction of controls is to reduce the consequence the frequency of occurrence of threats. Therefore, we propose an attribute; *AROR*, which measures the percentage of *ARO* mitigation.

$$ALE = SLE \times (ARO - AROR(\%)) \tag{24}$$

A set of extra functions (functions and constraints) are included in the feature models. These are listed in Appendix D. A pseudo-formal logic is employed to represent these constraints and functions. For instance, the formula of calculation of *ALE* is only established if the SSL/TLS feature is in the configuration.

$$\begin{aligned} SSL/TLS \rightarrow SSL/TLS\_ALE \\ = (SSL/TLS\_ARO - (SSL/TLS\_ARO \\ \times (SSL/TLS\_AROR))/100) \times SSL/TLS\_SLE) \end{aligned} \tag{25}$$

Nevertheless, the symbol $\neg$ is used to indicate the avoidance of a feature. For instance, the *AROR* for certificates has to be set to zero in the case of ignoring certificates.

$$\neg\ SSL/TLS.\text{Certificate} \rightarrow \text{Certificate\_}AROR = 0 \tag{26}$$

ALE for a *SSL*/*TLS* configuration is calculated by multiplying *ARO* minus AROR reduction and *SLE*. The *AROR* is defined depending on the *Protocol* and the infrastructure for *Authorization* selected (*Certificates* or *DigitalSignatures*). The *Cost* attribute is similarly established. Using these metrics, the *ROSI* of any configuration can be determined. On the other hand, other constraints have been stated, such as the constraint that limits *ALE* to at least 3000 monetary units/year, and the constraint that restricts the range of values for *AROR* from 20 to 30 in the case of protocol SSLv2.0. Thus, there are constraints to limit the values for attributes and others are stated to calculate functions related to these attributes. In the

Table 3
Description of features for SSL/TLS configuration.

Feature	Description
CipherSuite	Indicates the suites of key change method, cipher and message authentication code are supported
Key change method	Indicates that cryptographic algorithms have been employed to generate cryptographic keys
Cipher	Indicates that conventional cryptographic algorithms are employed to encrypt the message in the transmission
Message Authentication Code (MAC)	Indicates the algorithm employed to encrypt the message to provide integrity
Protocol	Indicates the version of protocol to be used
Session ID	Indicates the ID session established in the negation of connection client–server
Authentication method	Indicates the authentication method to be used (certificates or shared keys)
Digital signature	Indicates other types of signature support instead of certificates (SRP, PSK, or Anonymous)
Certificate	Indicates the type of certificates supported (×509 or OpenPGP)

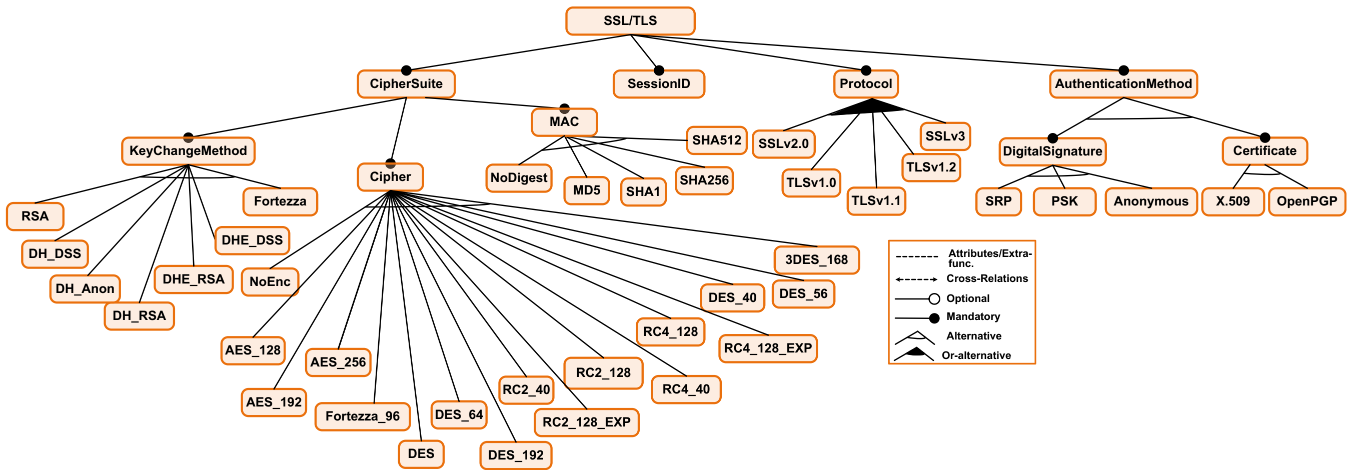


Fig. 13. Feature model of SSL/TLS.

Table 4
Description of attributes for feature model of SSL/TLS.

Attributes	Units	Range	Description
ARO	Times/year	Integer	Indicates the annual rate of occurrence per year
SLE	Monetary units	Integer	Indicates the single loss exposure
ALE	Monetary units/year	Integer	Indicates the annual loss exposure
AROR	%	[0–100]	Indicates the percentage of reduction of ARO
Cost	Monetary units	Integer	Indicates the cost of implementation

following section, a set of initial values is provided as a proof of concept to carry out a comparison of results as close to reality as possible. Nevertheless, the values of these metrics are generally established based on statistical data, reports, tracking processes and the expertise of professionals.

5.1.3. Feature model for authorization

Current business process notations, such as BPMN and UML diagrams, use a role-based perspective. Business process models can therefore define which specific roles are responsible for executing the activities represented in the business processes. Hence, users/customers enter the *BPMS* (authentication) taking a specific role, and are enabled to execute the business processes defined for this role. Once users are authenticated (authentication) it is necessary to ensure these users only have access to information and business processes for which they are authorized (authorization). Traditionally, authorization is established by means of filtering mechanisms [46]. Filtering mechanisms consists of the specification of security policies (such as *RBAC* policies) that define conditions (rules) under which users can or cannot access the resources (systems or information). In recent years, Web Application Firewalls (WAF) have emerged which block unauthorized access [47]. These mechanisms inspect the contents of traffic (HTTP traffic), and block specified content, such as access to certain websites, or attempts to exploit known logical vulnerabilities (web attacks by *SQL injections* or *Cross-Site Scripting* (XSS) [48]).

In general, WAF systems are located as full and transparent proxy. In our case, a WAF can be located between the *BPMS* and other systems and are responsible for the inspection of the content of the request from and response to the *BPMS* such as shown in Fig. 14.

WAFs might be installed as part of the IT infrastructure by means of appliance systems installed in front of BPMS or as an add-on of the *BPMS*. Various open-source WAF solutions exist: *TrustWaveModSecurity*® [49], *QualysIronBee* [50], and *OWASPESAPIJava-WAF* [51]. Furthermore, other commercial solutions are available, such as *AQTronixWebKnight* [52], *StingrayApplicationFirewall* [53], *WebCastellum* [54], *ZionSecured* [55], *Guardian@JUMPERZ.NET* [56], and *EasyWAF* [57].

WAFs are stated by policies that are composed of a set of rules. If any content matches any rule, then an action is triggered and the web traffic goes through the WAF. Rules can be customized in order to detect common web attacks (typical code injections), malicious web content, malicious file upload, violation of protocols, and Denial of Service attacks (DoS), among others. A summary of the main features for the configuration of rules in WAFs is given in Table 5.

The specification of rules presents a highly intricate casuistry. Rules may be defined with regard to certain web content, such as files, response, request information, environment information; operators, such as validation of schemas, verification of schemas, detection of regular expressions; actions, such as allow, block, log content; among others. This complexity (cf. Fig. 15) requires a high level of knowledge and expertise of the functionalities of the WAF. We have defined a feature model (SC^{az}) of the most common WAF rule features. This feature model gives useful information for security experts in order to obtain/analyze configurations in accordance with specific and necessary features.

We have developed an analysis of the most widely used WAF (see Appendix C). As a result of this analysis, a feature model has been defined as shown in Fig. 16. This feature model shows only certain features since the representation of all features would render the diagram impossible to decipher.

In this section, various feature models (SC^{c}, SC^{a}, SC^{at} and SC^{az}) have been presented relative to security tools that enable the achievement of security goals for *BPMS*. As mentioned earlier, feature models are to be analyzed automatically by means of feature-oriented model analysis (FODA) based on constraint programming techniques. In the following section, the analysis carried out in order to attain the best configurations is presented.

5.2. Automatic and optimal selection of IT security configurations

In order to automate the selection and generation of optimal configurations, feature model analysis based on Constraint Programming (CP) techniques is used. Feature models have therefore been transformed into Constraint Satisfaction Problems (CSP) following the approach [34,35] explained in Section 4. Feature models have been implemented in *COMET*® [58]. *COMET*® is a very powerful constraint solver with features for optimized searches. To the best of our knowledge, *COMET*® has never been used for any current feature model analysis tool, such as *FamaSuite* [59], *VariaMos* [60], *and pure*::*variants* [61]. The constraint programmes used in this section are available for evaluation and downloading at [13].

The first analysis consists of the identification of the total number of configurations. The result for this analysis is given in Table 6. The analysis indicates: *number of features*, *relations* (mandatory, optional, *XOR* and *OR*), *void feature model* [34] is a model validation operation that indicates whether a feature model is void (indicated by •) or not (indicated by ×). A feature model is void if it represents no products. The reasons that may make a feature model void are related to a wrong usage of cross–tree constraints. *Number of configurations* and *time of performance* to obtain the configurations. For this analysis, the (exhaustive) default search provided by the constraint solver is applied.

The number of configurations represents the number of valid products that are achieved with this FM according to SPL theory. 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*. Nevertheless, the number of configurations for fault tolerance remains very low. 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, feature models can be extended with attributes and extra functionalities in order to adjust the searches. In this case, extended feature models might be transformed into Constraint Optimization Problems (COP) [35]. A COP searches for a solution in accordance with an optimization function such as that defined in Definition 6. Thus, COP strives to find the best 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 uses 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 could consider the attributes of security level, and it is interested in configurations whose security level is high. Therefore, the COP aim would be to search the best configuration with a high level of security.

In real scenarios, organizations require controls to be selected according to multiple criteria. Therefore, various objective functions are used, for instance cost minimization, and the maximization of the use of resources. In this case, the problem is to determine the configuration based on multiple objectives such as those defined in Definition 7. Occasionally, the objectives may be weighted in order to grant one objective a higher priority than the others. This kind of search is called a weighted search and is a specialization of a multiple objective search. In the revised literature, the authors in [62,63] propose a multi-objective approach for the optimal selection of controls. In [64], the authors propose the selection of ISO/IEC 27001 [63] controls based on multiple objectives, (cost, benefit, etc.). Nevertheless, Neubauer and Heurix merely indicate that selection is carried out using search-based techniques, and that there are no implementations. In our approach, the generation of optimal configurations are proposed based on multiple objectives by means of Multi-objective COP

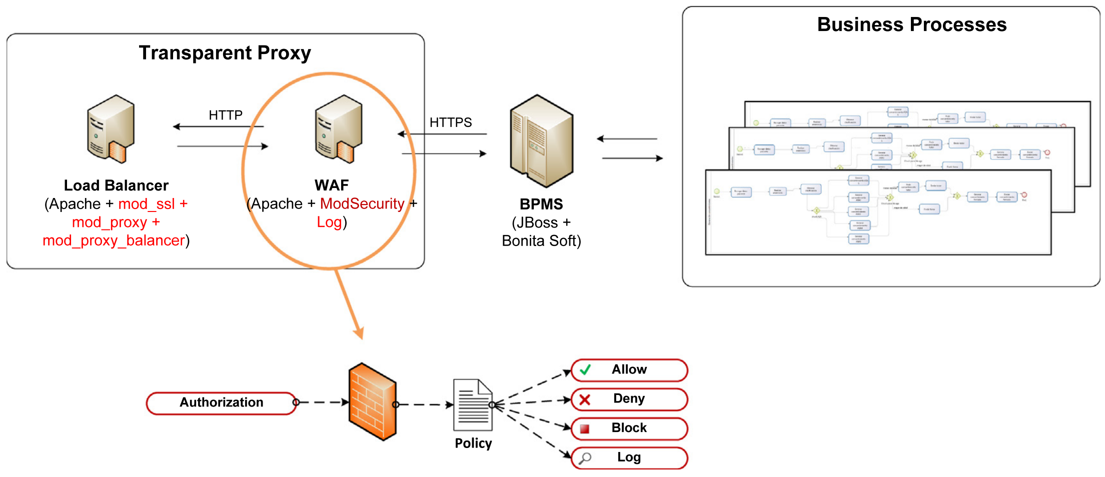


Fig. 14. Scenario for authorization enforcement in BPMS.

```
SecMarker BEGIN_HOST_CHECK

   SecRule&REQUEST_HEADERS:Host "@eq 0" \

   "skipAfter:END_HOST_CHECK,phase:2,rev:'2.2.4',t:none,block,msg:'Request
   Missing                              a                              Host
   Header',id:'960008',tag:'PROTOCOL_VIOLATION/MISSING_HEADER_HOST',tag:'WASCTC/W
   ASC-
   21',tag:'OWASP_TOP_10/A7',tag:'PCI/6.5.10',severity:'5',setvar:'tx.msg=%{rule.
   msg}',setvar:tx.anomaly_score=+%{tx.notice_anomaly_score},setvar:tx.protocol_v
   iolation_score=+%{tx.notice_anomaly_score},setvar:tx.%{rule.id}-
   PROTOCOL_VIOLATION/MISSING_HEADER-%{matched_var_name}=%{matched_var}"

   SecRuleREQUEST_HEADERS:Host "^$" \

   "phase:2,rev:'2.2.4',t:none,block,msg:'Empty                            Host
   Header',id:'960007',tag:'PROTOCOL_VIOLATION/MISSING_HEADER_HOST',severity:'5',
   setvar:'tx.msg=%{rule.msg}',setvar:tx.anomaly_score=+%{tx.notice_anomaly_score
   },setvar:tx.protocol_violation_score=+%{tx.notice_anomaly_score},setvar:tx.%{r
   ule.id}-PROTOCOL_VIOLATION/MISSING_HEADER-%{matched_var_name}=%{matched_var}"
```

Fig. 15. Example of rules for the detection of protocol anomalies in ModSecurity.

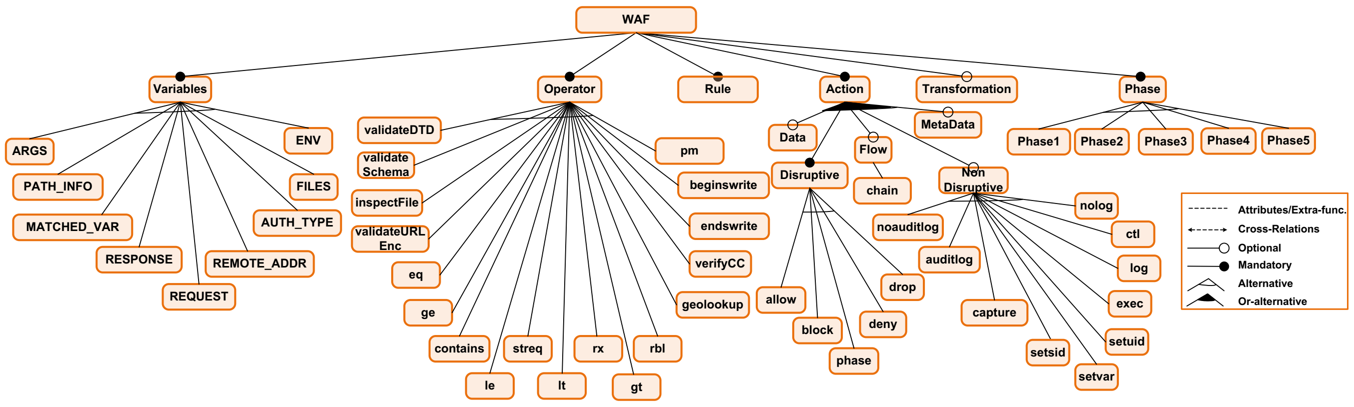


Fig. 16. Overview of the feature model for WAF.

Table 5
Description of features for WAF configuration.

Feature	Description
Variable	Indicates the data fields against apply the operations
Operator	Indicates the types of operations to perform against the variables
Actions	Indicates the type of action to execute in the case of matching
Transformation	Indicates the transformation to apply over data before to apply operators
Phase	Indicates when the rule is applicable

(MCOP) [65]. In this case, COPs are adapted to attain multiple objectives. To the best of our knowledge no contributions exist in feature-oriented model analysis towards obtaining configurations based on multi-objectives.

Our implementations of feature models with attributes and extra functionalities have been extended in order to generate configurations according to certain optimization criteria. The extension provides information relative to metrics (M), such as that indicated in the formalization. Therefore, we have two options for the selection (Fig. 17): (1) one that provides all possible configurations and (2) one that provides the best configurations according to certain optimization criteria (in the formalization denoted as F).

In order to compare the results of the determination of configuration with and without optimization, a comparative analysis is given in Table 7, which assumes that the constraints and functions are included in the feature models. The comparative study has been performed in two phases: (1) a search using single objectives (Definition 6) and (2) a search using multi-objectives (Definition 7). The table shows information on the *attributes* included; *optimization function* used for each case; *number of configurations* achieved; and the performance given in milliseconds.

As shown in Appendix D, the *ALE* is calculated by the combination of *ARO*, *AROR* 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

Table 6
Results of feature model analysis.

Feature Model (FM)	Number of features	Mandatory	Optional	XOR	Or	Void feature model	Number of configurations	Time (ms)
Fault Tolerance (FT)	17	8	1	7	0	×	7	9
SSL/TLS	49	10	0	42	5	×	3683	4699
WAF	62	6	6	57	4	×	241,920	77,427

fits the minimum of Cost and *ALE* due to the over-constrained problem. In this case, there are two options: (1) accept that there are no solutions and (2) relax some of the objectives in order to find solutions close to the optimum. A multi-objective search for *SSL*/*TLS* has been performed which relaxes the objectives (marked by the symbol $\sim$). For a better understanding, a graphic of the *ALE* calculation is represented in Fig. 18. In our initial tests, *SLE* and *ARO* hold a fixed value, and the *AROR* is calculated in terms of the configuration selected. In this case, it can be observed how the minimum has been tightened in eight iterations.

It should be pointed out that the numbers of configurations are high due to the combination introduced by the attributes and extra functionalities. An example of configurations obtained for the *SSL/TLS* example is shown in Table 8. This table shows different configurations achieved for the multi-objective of *minimize ALE* and the relaxing of *Cost*. This configuration can help security stakeholders to deal with decision making regarding security configurations. However, these configurations give an overview of the space of configurations and even specific configurations can be customized with respect to certain multi-objectives. Nevertheless, a large number of configurations are remain inoperative for security administrators; it should be interesting, for example, to introduce an ordering criterion in order to generate a list of the best configurations according to the tightened objectives, such as indicated in the # column.

Regarding fault tolerance results, it can be observed that good results are obtained for single-objective searches. In the case of multi-objectives, these solutions are achieved very fast since there are no dependencies between the objectives. Therefore, the constraint solver does not need to find Pareto-efficient solutions since the optimum is determined independently. In Table 9, several configurations are given; the solutions that match with the optimum configurations are highlighted.

5.3. Discussion of results

In the previous section, we performed different analyses on the proposed feature models. The first analysis consisted of obtaining the total number of configurations regardless of attributes and extra functionalities. In addition, a model consistency operation has been applied to the models in order to detect possible void feature models. The second analysis consisted of applying an optimized search in order to achieve specific configurations with regard to one or multiple objective functions. The aim of these analyses is to facilitate the tasks of different actors involved in business process design when selecting IT security configurations.

Regarding the first analysis in the absence of optimized searches (cf. Table 6), the number of configurations of SSL/TLS is very high in comparison with fault tolerance due to the combinatory nature of the feature models presented.

In the second analysis regarding optimized searches (cf. Table 7), we observe that the number of configurations is increased in both cases in the absence of optimized searches. This is caused by the combinatory nature of attributes and extra constraints introduced in the feature models.

The performance (time in milliseconds) of searches is very low in all analyses.

The usefulness of the results obtained in the selection of IT configurations can be observed from different points of views:

- For Business and security managers, our approach enables the analysis and inference of IT configurations to determine whether certain settings are more effective against certain objectives such as cost, ALE, ROSI, Risk Reduction, etc.
- For Business and security administrators, our approach enables them to obtain blueprints or templates to configure systems. Furthermore, our approach can ensure that templates suit the organization requirements.

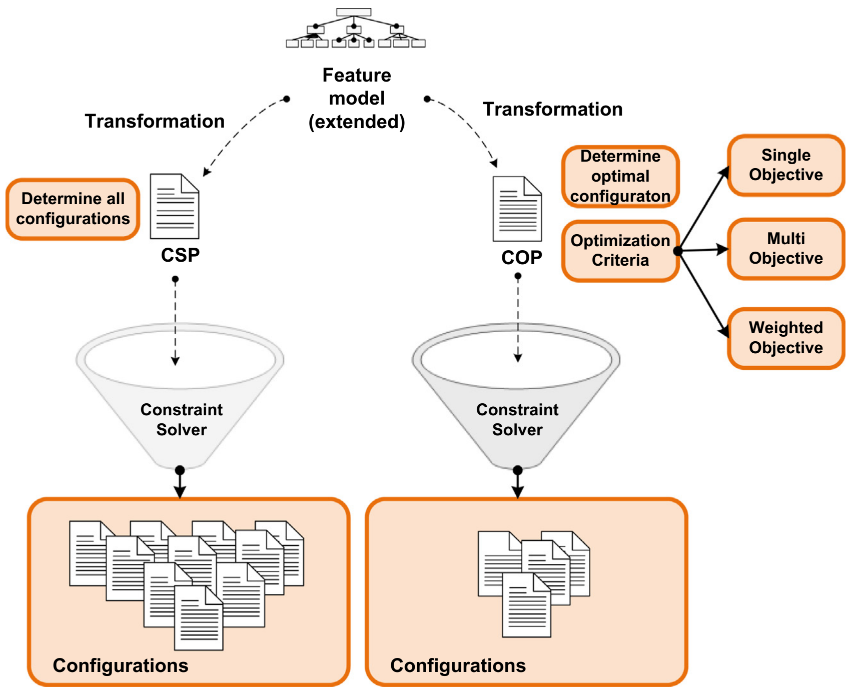


Fig. 17. Process of selection of configurations.

Table 7
Results of analysis of generation of configurations with attributes.

Feature model	Optimization criteria	Number of configurations	Time (ms)
SSL/TLS	Single Objective: Minimize (ALE)	13,138	2041
	Single Objective: Maximize (AROR)	5268	1255
	Single Objective: Minimize (Cost)	1800	2394
	Multi-Objective: Maximize (AROR) + Minimize (ALE)	5268	5257
	Multi-Objective: Minimize (Cost) + Minimize (ALE)	0	406
	Multi-Objective: ~Minimize (Cost) + Minimize (ALE)	108	880
Fault tolerance	Single Objective: Minimize (MTTR)	4	39
	Single Objective: Maximize (Risk Reduction)	58	42
	Multi-Objective: Minimize (MTTR) + Maximize (Risk Reduction)	36	39

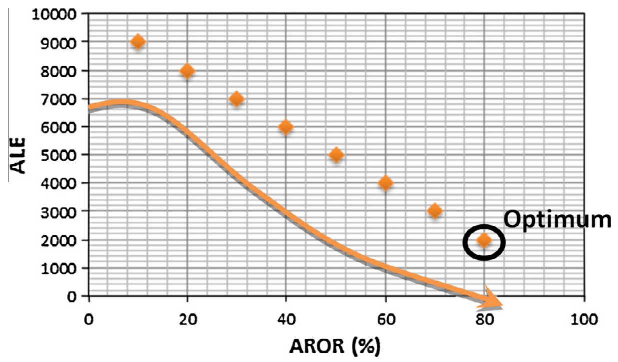


Fig. 18. Constraint solver iterations for ALE and AROR determination.

- In both cases, due to the easy-to-modify characteristic of feature models and objectives and the provisioning of inference, reasoning techniques enable a quick, automatic, flexible and agile attainment of IT security configurations.

The major advantage of our approach from both points of view is the automation of the selection process, and low execution time for searches. Thus, a task such as configuring a system that is typically time-consuming and error-prone can be reduced to 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 feature model as complete as possible. Nevertheless, once feature models are defined it only requires an update cycle.

6. Conclusions and future research

In this paper, the issue of the automatic and optimal selection of configurations of controls for current *BPMS* has been tackled. The main obstacles in this system 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 controls present a big challenge in *BPMS* for several reasons: (1) it is a human, manual, time-consuming, and error-prone task; (2) it involves many security stakeholders such as business analysts, security managers, and administrators; and (3) 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 enable the automatic selection of security configurations in accordance with the needs of the organization.

A large proportion of the literature revised herein proposes an integration of risk information by extending graphical business process models in order to aid in the documentation of business process risk assessments. Nevertheless, these approaches fail to take into consideration the integration of any mechanism to determine potential controls. Regarding the generation of configurations, there are various authors who have enabled the specification of security in business process models although they focused on the authorization in a particular environment. Several studies apply feature model analysis for the generation of secure feature models. On the other hand, there exist approaches that determine configurations in accordance with certain requirements in the quality of service. These approaches, however, use only a qualitative approach.

Table 8
Configurations of SSL/TLS for optimization of ALE and Cost.

#	Digital signature			Certificate		CipherSuite			Protocol	Objective	
	PSK	SRP	Anon.	X509	OpenPGP	Key change method	CipherEnc	MAC		ALE	Cost
1				√		RSA			TLSv1.0	2.000	45
2				√		RSA		MD5	TLSv1.0	2.000	45
3				√		RSA	IDEA-128	SHA-1	TLSv1.1	2.000	50
4				√		Fortezza		SHA-256	TLSv1.1	2.000	50
5				√		DHE_RSA	3DES 168	SHA-1	TLSv1.1	2.000	50

Table 9
Configurations of fault tolerance for optimization of MTTR and risk reduction.

#	Dynamic binding				NVP		Check pointing			Objectives	
	Number of replicas	Oracle	Binder	Binder backup	Number of variants	Adjudicator	Number of Sensors	Oracle	Number of check points	MTTR	Risk reduction (%)
1					5 Variants 10 ms	Median 10 ms				20 ms	80
2					7 Variants 10 ms	Consensus 10 ms				20 ms	80
3					9 Variants 10 ms	Consensus 10 ms				20 ms	80
4							2 Replicas 10 ms	45 ms	15 ms	70 ms	80
5	10 Replicas 10 ms	30 ms	10 ms	10 ms						50 ms	70

In this paper, we propose automation of the selection of configurations by means of feature models and constraint programming techniques. To this end, we first provide a catalogue of feature models based on certain IT security controls; this catalogue could be integrated into *BPMS*, to treat typical IT security risks with respect to confidentiality, integrity, authentication and authorization, and availability. These feature models ascertain which characteristics must support the *BPMS* in order to achieve security objectives. In order to automate the generation of configurations, feature model analysis mechanisms have been used. Feature models have been modeled through constraint programs that enable information about configurations to be inferred. This analysis enables us to ascertain, for example, how many configurations are possible, and which configuration is the best option according to attributes and functions included in the model; it is even possible to ascertain whether a configuration is valid or not. Furthermore, these mechanisms provide an agile selection of configurations by means of including new attributes and functions. These attributes and functions enable the selection of configurations in accordance with various objectives. We have improved the selection of configuration by providing new search criteria based on a single objective and on multi-objectives. There exist various studies into the selection of controls based on multi-objectives, however, these studies are focused on the selection of controls, and not on the provision of the generation of configurations for controls.

The present work can be extended in several ways. Risk dependencies can be studied in order to improve the selection of controls. We propose that OPBUS tools be extended with feature model analysis capabilities. Thus, feature models can be integrated as profiles of configuration of business processes, and, through the use of feature analysis, configurations with regard to the risk problems identified can be inferred. On the other hand, feature models can be employed to describe characteristics in security patterns that enable the profile of security configurations to be represented with respect to typical security threats and vulnerabilities. Furthermore, a study into the feature model analysis capabilities could be performed in order to provide new functions in the selection of security configurations.

Acknowledgements

This work has been partially funded by the Department of Innovation, Science and Enterprise of the Regional Government of Andalusia (under Grant P08-TIC-04095), by the Spanish Ministry of Science and Education (under Grant TIN2009-13714), and by FEDER (under the ERDF Program). The authors would like to thank Prof. Christoph Meinel, and the Hasso-Plattner-Institute for their valuable provision of information. The authors would like to thank to Fidetia Foundation for supporting.

Appendix A. Feature analysis of SSL and TLS protocol

Transport Layer Security (TLS) protocol and Secure Socket Layer (SSL) (hereinafter SSL/TLS) enable the authentication of client and server by providing signatures (certificates or passwords). Furthermore, data might be signed and ciphered in order to achieve data confidentiality and data integrity.

SSL/TLS provides three main security services: confidentiality, by encrypting data; message integrity, by using a message authentication code (MAC); authentication, by using digital signatures. SSL/TLS allows the authentication of both parties, server authentication with an unauthenticated client, and total anonymity. The authentication of client and server might be carried out through digital signatures. Nowadays, these digital signatures are mostly based on certificates (i.e. X.509 standard) or shared keys. Certificates are based on the concept of public key cryptography known as asymmetric cryptography. This method generates two keys: a public key and private key. If one key is used to encrypt a message (signed and ciphered) then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key). Anyone may encrypt a message using the public key, but only the owner of the private key will be able to decrypt and read it. In the case of using certificates, these always have to be verified to ensure proper signing by a trusted Certificate Authority (CA). On the other hand, these protocols also provide anonymous authentication by using Diffie-Hellman for key exchange from SSLv3.0, TLSv1.0 and earlier versions.

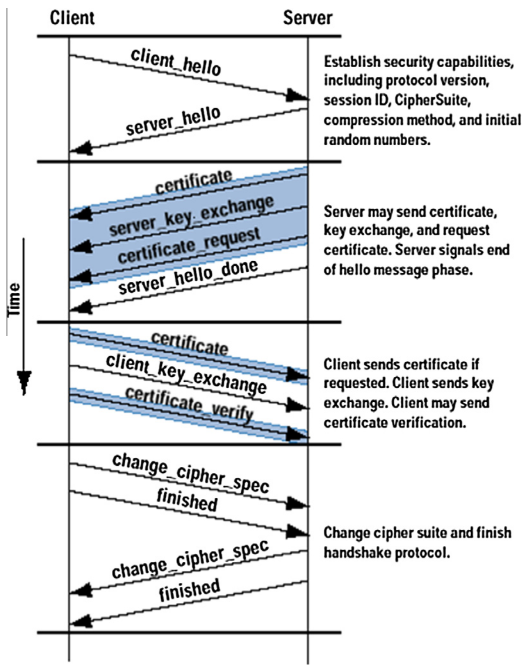

Client
Server
client_hello
server_hello
Establish security capabilities, including protocol version, session ID, CipherSuite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Change cipher suite and finish handshake protocol.

Fig. 19. SSL handshake protocol actions [71].

SSL/TSL protocol is based on a handshake sequence (Fig. 19) whose main features, as used by the client and server, are listed below:

1. Negotiate the **Cipher Suite** to be used during data transfer, and exchange random numbers (master key).
2. Establish and share a **Session ID** between client and server.
3. Optionally **authenticate** the server to the client.
4. Optionally **authenticate** the client to the server.

The first step, cipher suite negotiation, allows the client and server to choose a **Cipher Suite**[2] supported by both of them. For instance, the SSLv3.0 protocol specification defines 31 Cipher Suites. A Cipher Suite is defined by the following components:

- **Key Exchange Method** defines which cryptographic algorithms have been used to generate cryptographic keys. For instance, SSLv2.0 uses RSA key exchange only, while SSLv3.0 supports a choice of key exchange algorithms including the RSA key exchange when certificates are used, and Diffie-Hellman key exchange for exchanging keys without certificates and without prior communication between client and server.

[2] The *CipherSuite* supported by each version protocol can be consulted in [32,33,53,59,60].

Table 10
Summarization of feature related to SSL/TLS.[a]

Protocol	Key Exchange Method (KEM)	Cipher (C)	Message Authentication Code (MAC)	Master key	Key-size limit	Session ID	Constraints
SSLv2.0	RSA	RC4 128 bits	MD5	256 bits	Minimum 512 bits	16 bytes	Authentication by certificate only RSA and MD5
		RC4 128 bits Export					
		RC2 128 bits					
		RC2 128 bits Export					
		IDEA 128 bits					
		DES 64 bits					
		DES 192 bits EDE3					
SSLv3.0	**Null**	**Null**	**Null**	384 bits	Minimum 512 bits	32 bytes	Combination of KEM-C-MAC methods are listed in Appendix 6 of RFC 6101 [70]
	RSA	**RC4 40 bits**	MD5				
	DH[b] with DSS	RC4 128 bits	**SHA**				
	DH with RSA	RC4 128 bits Export					
	DHE[c] with DSS	**RC2 40 bits**					
	DHE with RSA	RC2 128 bits					
	DH with Anonymous	RC2 128 bits Export					
	Fortezza	IDEA 128 bits					
		DES 56 bits					
		DES 64 bits					
		DES 192 bits EDE3					
		DES 40 bits					
		Fortezza 96 bits					
		3DES 168 bits					
TLSv1.0	Null	Null	Null	384 bits	Minimum 512 bits	32 bytes	Combination of KEM-C-MAC methods are listed in Appendix 5.A of RFC 2246 [42]
	RSA	RC4 40 bits	MD5				
	DH with DSS	RC4 128 bits	SHA				
	DH with RSA	RC2 40 bits					
	DHE with DSS	IDEA 128 bits					
	DHE with RSA	DES 56 bits					
	DH with Anonymous	DES 40 bits					
		3DES 168 bits					
TLSV1.1	Null	Null	Null	384 bits	None	32 bytes	Combination of KEM-C-MAC methods are listed in Appendix A.5 of RFC 4346 [68]
	RSA	RC4 128 bits	MD5				
	KRB5	RC2 40 bits	SHA				
	DH with DSS	IDEA 128 bits					
	DH with RSA	DES 56 bits					
	DHE with DSS	DES 40 bits					
	DHE with RSA	3DES 168 bits					
	DH with Anonymous	**AES 128 bits**					
		AES 256 bits					
TLSv1.2	Null	Null	Null	384 bits	None	32 bytes	Combination of KEM-C-MAC methods are listed in Appendix A.5 of RFC 5246 [69]
	RSA	RC4 128 bits	MD5				
	RSA PSK[d]	3DES 168 bits	SHA				
	DH with DSS	AES 128 bits	**SHA224**[d]				
	DH with RSA	AES 256 bits	**SHA256**				
	DHE with DSS		**SHA384**[d]				
	DHE with RSA						
	ECDH with EDSA[d]						
	ECDH with RSA[d]						
	DH with Anonymous						

[a] TLS v1.x is compatible with previous versions of SSL and TLS.
[b] Diffie-Hellman signed with DSS certificates.
[c] Ephemeral Diffie-Hellman signed with DSS certificates.
[d] These features are only available as an extension in order to indicate the signature and hash algorithm.

- **Cipher for Data Transfer** defines which conventional cryptographic algorithms are used to encrypt the message in the transmission. For instance, SSLv2.0 supports a reduced number of ciphers such as RC2, RC4 and Data Encryption Standard (DES). However, SSLv3.0 and earlier versions of TLS increase the number of cryptographic algorithms supported.

- **Message Digest** is used to create a **Message Authentication Code (MAC)** which is encrypted with the message to provide integrity and to prevent replay attacks. For instance, SSLv2.0 only supports MD5, while SSLv3.0 and earlier versions of TLS support other methods such as SHA-1 (see Table 10).

Appendix B. Cross-tree constraint for feature model of SSL/TLS

This appendix provides the cross-tree constraints that cannot be represented in the FM of SSL/TLS. The syntax used for the representation of the constraints corresponds with *COMET*® notation.

```
// SSLv2
manager.post((protocol == SSLv2 && certificate == 1) ~>
(keychangemethod == RSA && x509 == 1 && mac == MD5 &&
(cipher_enctype == NoEnc ||
cipher_enctype == RC4128 ||
cipher_enctype == DES192 ||
cipher_enctype == RC2128 ||
cipher_enctype == DES64 ||
cipher_enctype == RC4128EXP ||
cipher_enctype == RC2128EXP)));
// SSLv3 // RSA and certificate
manager.post((protocol == SSLv3  &&keychangemethod == RSA) ~>
(cipher_enctype == NoEnc&&mac == MD5 ||
cipher_enctype == RC440 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == SHA1 ||
cipher_enctype == RC240 &&mac == MD5 ||
cipher_enctype == IDEA128 &&mac == SHA1 ||
cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
// DH and certificate
manager.post((protocol == SSLv3 &&keychangemethod == DH_DSS) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == SSLv3 &&keychangemethod == DH_RSA) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == SSLv3 &&keychangemethod == DHE_DSS) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == SSLv3 &&keychangemethod == DHE_RSA) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
// DH anonymous
manager.post((protocol == SSLv3 &&keychangemethod == DH_anon) ~>
(cipher_enctype == RC440 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
// Fortezza PSK or SRP
manager.post((protocol == SSLv3 &&keychangemethod == Fortezza) ~>
(cipher_enctype == NoEnc&&mac == SHA1 ||
cipher_enctype == Fortezza&&mac == SHA1 ||
cipher_enctype == RC4128 &&mac == SHA1))
// TLS // RSA and certficates
manager.post((protocol == TLSv11 &&keychangemethod == RSA) ~>
(cipher_enctype == NoEnc&&mac == MD5 ||
cipher_enctype == NoEnc&&mac == SHA1 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == SHA1 ||
cipher_enctype == RC240 &&mac == MD5 ||
cipher_enctype == IDEA128 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv1 &&keychangemethod == RSA) ~>
(cipher_enctype == NoEnc&&mac == MD5 ||
cipher_enctype == NoEnc&&mac == SHA1 ||
cipher_enctype == RC440 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == SHA1 ||
cipher_enctype == RC240 &&mac == MD5 ||
cipher_enctype == IDEA128 &&mac == SHA1 ||
cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv12 &&keychangemethod == RSA) ~>
(cipher_enctype == NoEnc&&mac == MD5 ||
cipher_enctype == NoEnc&&mac == SHA1 ||
cipher_enctype == NoEnc&&mac == SHA256 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256));
// DH and certficates
manager.post((protocol == TLSv11 &&keychangemethod == DH_DSS) ~>
(cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv11 &&keychangemethod == DH_RSA) ~>
(cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv11 &&keychangemethod == DHE_DSS) ~>
(cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv11 &&keychangemethod == DHE_RSA) ~>
(cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv1 &&keychangemethod == DH_DSS) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv1 &&keychangemethod == DH_RSA) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv1 &&keychangemethod == DHE_DSS) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv1 &&keychangemethod == DHE_RSA) ~>
(cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv12 &&keychangemethod == DH_DSS) ~>
(cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv12 &&keychangemethod == DH_RSA) ~>
(cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv12 &&keychangemethod == DHE_DSS) ~>
(cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
manager.post((protocol == TLSv12 &&keychangemethod == DHE_RSA) ~>
(cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256 ||
cipher_enctype == TripleDES168 && mac == SHA1 ));
// DH anonymous
manager.post((protocol == TLSv11 &&keychangemethod == DH_anon) ~>
(cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv1 &&keychangemethod == DH_anon) ~>
(cipher_enctype == RC440 &&mac == MD5 ||
cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == DES40 &&mac == SHA1 ||
cipher_enctype == DES &&mac == SHA1 ||
cipher_enctype == TripleDES168 &&mac == SHA1));
manager.post((protocol == TLSv12 &&keychangemethod == DH_anon) ~>
(cipher_enctype == RC4128 &&mac == MD5 ||
cipher_enctype == TripleDES168 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA1 ||
cipher_enctype == AES256 &&mac == SHA1 ||
cipher_enctype == AES128 &&mac == SHA256 ||
cipher_enctype == AES256 &&mac == SHA256));
```

Appendix C. Feature analysis of Web Application Firewall (WAF) systems

WAFs are stated by means of rules. One rule specified, for instance, by *ModSecurity* and *IronBee* has the following structure: Rule VARIABLES OPERATOR [ACTIONS]. Furthermore, rules could specify a set of operations (*Operations*/*Transformation*) used to inspect the content. In the tables below, the different possibilities that can be used in the definition of a rule are described. The analysis is focused on *ModSecuriyt*, *Ironbee*, and *ESAPIWAF* systems.

Name	OS	Language	Variables
TrustWaveModSecurity	Unix, Windows	Independant	ARGS, ARGS_COMBINED_SIZE, ARGS_GET, ARGS_GET_NAMES, ARGS_NAMES, ARGS_POST, ARGS_POST_NAMES, AUTH_TYPE, DURATION, ENV, FILES, FILES_COMBINED_SIZE, FILES_NAMES, FILES_SIZES, FILES_TMPNAMES, GEO, HIGHEST_SEVERITY, INBOUND_ERROR_DATA, MATCHED_VAR, MATCHED_VARS, MATCHED_VAR_NAME, MATCHED_VARS_NAMES, MODSEC_BUILD, MULTIPART_CRLF_LF_LINES, MULTIPART_STRICT_ERROR, MULTIPART_UNMATCHED_BOUNDARY, PATH_INFO, PERF_COMBINED, PERF_GC, PERF_LOGGING, PERF_PHASE1, PERF_PHASE2, PERF_PHASE3, PERF_PHASE4, PERF_PHASE5, PERF_SREAD, PERF_SWRITE, QUERY_STRING, REMOTE_ADDR, REMOTE_HOST, REMOTE_PORT, REMOTE_USER, REQBODY_ERROR, REQBODY_ERROR_MSG, REQBODY_PROCESSOR, REQUEST_BASENAME, REQUEST_BODY, REQUEST_BODY_LENGTH, REQUEST_COOKIES, REQUEST_COOKIES_NAMES, REQUEST_FILENAME, REQUEST_HEADERS, REQUEST_HEADERS_NAMES, REQUEST_LINE, REQUEST_METHOD, REQUEST_PROTOCOL, REQUEST_URI, REQUEST_URI_RAW, RESPONSE_BODY, RESPONSE_CONTENT_LENGTH, RESPONSE_CONTENT_TYPE, RESPONSE_HEADERS, RESPONSE_HEADERS_NAMES, RESPONSE_PROTOCOL, RESPONSE_STATUS, RULE, SCRIPT_BASENAME, SCRIPT_FILENAME, SCRIPT_GID, SCRIPT_GROUPNAME, SCRIPT_MODE, SCRIPT_UID, SCRIPT_USERNAME, SERVER_ADDR, SERVER_NAME, SERVER_PORT, SESSION, SESSIONID, STREAM_INPUT_BODY, STREAM_OUTPUT_BODY, TIME, TIME_DAY, TIME_EPOCH, TIME_HOUR, TIME_MIN, TIME_MON, TIME_SEC, TIME_WDAY, TIME_YEAR, TX, UNIQUE_ID, URLENCODED_ERROR, USERID, WEBAPPID, WEBSERVER_ERROR_LOG, XML
QualysIronBee	Unix	Independant	ARGS, AUTH_PASSWORD, AUTH_TYPE, AUTH_USERNAME, CAPTURE, FIELD, FIELD_NAME, FIELD_NAME_FULL, GEOIP, HTP_REQUEST_FLAGS, HTP_RESPONSE_FLAGS, LAST_MATCHED, REMOTE_ADDR, REMOTE_PORT, REQUEST_BODY, REQUEST_BODY_PARAMS, REQUEST_CONTENT_TYPE, REQUEST_COOKIES, REQUEST_FILENAME, REQUEST_HEADERS, REQUEST_HOST, REQUEST_LINE, REQUEST_METHOD, REQUEST_PROTOCOL, REQUEST_URI, REQUEST_URI_FRAGMENT, REQUEST_URI_HOST, REQUEST_URI_PARAMS, REQUEST_URI_PASSWORD, REQUEST_URI_PATH, REQUEST_URI_PORT, REQUEST_URI_RAW, REQUEST_URI_SCHEME, REQUEST_URI_QUERY, REQUEST_URI_USERNAME, RESPONSE_BODY, RESPONSE_CONTENT_TYPE, RESPONSE_COOKIES, RESPONSE_HEADERS, RESPONSE_LINE, RESPONSE_MESSAGE, RESPONSE_PROTOCOL, RESPONSE_STATUS, SERVER_ADDR, SERVER_PORT, SITE_NAME, TX, UA
OWASP ESAPI Java-WAF[a]	Independant	XML	Request.parameters.some_parameter, request.headers.some_header, request.uri, request.url

[a] *Valid only for J2EE applications.*

Name	Operators	Actions	Transformation
TrustWaveModSecurity	beginsWith, contains, endsWith, eq, ge, geoLookup, gsbLookup, gt, inspectFile, ipMatch, le, lt, pm, pmf, pmFromFile, rbl, rsub, rx, streq, strmatch, validateByteRange, validateDTD, validateSchema, validateUrlEncoding, validateUtf8Encoding, verifyCC, verifyCPF, verifySSN, within	Allow, append, auditlog, block, capture, chain, ctl, deny, deprecatevar, drop, exec, expirevar, id, initcol, log, logdata, msg, multiMatch, noauditlog, nolog, pass, pause, phase, prepend, proxy, redirect, rev, sanitiseArg, sanitiseMatched, sanitiseMatchedBytes, sanitiseRequestHeader, sanitiseResponseHeader, severity, setuid, setsid, setenv, setvar, skip, skipAfter, status, t, tag, xmlns	Base64decode, sqlHexDecode, base64DecodeExt, base64Encode, cmdLine, compressWhitespace, cssDecode, escapeSeqDecode, hexDecode, hexEncode, htmlEntityDecode, jsDecode, length, lowercase, md5, none, normalisePath, normalisePathWin, parityEven7bit, parityOdd7bit, parityZero7bit, removeNulls, removeWhitespace, replaceComments, removeCommentsChar, removeComments, replaceNulls,

(*continued on next page*)

Appendix C (*continued*)

Name	Operators	Actions	Transformation
			urlDecode, urlDecodeUni, urlEncode, sha1, trimLeft, trimRight, trim
QualysIronBee	Contains, dfa, eq, ge, gt, ipmatch, le, lt, ne, pm, pmf, rx, streq	Allow, logdata, block, capture, chain, confidence, delRequestHeader, delResponseHeader, id, msg, phase, rev, setflag, setRequestHeader, setResponseHeader, setvar, severity, status, t, tag	Base64decode, compressWhitespace, count, htmlEntityDecode, length, lowercase, removeWhitespace, removeComments, replaceComments, trim, trimLeft, trimRight, urlDecode, min, max, normalize, path
OWASP ESAPI Java-WAF	Equals, exists, inList, Contains. Pattern	Redirect, block, none, log	~

Name	Phase	Type of rules
TrustWaveModSecurity	Request headers request body, response headers, response body, logging	SecAction, SecArgumentSeparator, SecAuditEngine, SecAuditLog, SecAuditLog2, SecAuditLogDirMode, SecAuditLogFileMode, SecAuditLogParts, SecAuditLogRelevantStatus, SecAuditLogStorageDir, SecAuditLogType, SecCacheTransformations, SecChrootDir, SecComponentSignature, SecContentInjection, SecCookieFormat, SecDataDir, SecDebugLog, SecDebugLogLevel, SecDefaultAction, SecDisableBackendCompression, SecGeoLookupDb, SecGsbLookupDb, SecGuardianLog, SecHttpBlKey, SecInterceptOnError, SecMarker, SecPcreMatchLimit, SecPcreMatchLimitRecursion, SecPdfProtect, SecPdfProtectMethod, SecPdfProtectSecret, SecPdfProtectTimeout, SecPdfProtectTokenName, SecReadStateLimit, SecWriteStateLimit, SecRequestBodyAccess, SecRequestBodyInMemoryLimit, SecRequestBodyLimit, SecRequestBodyNoFilesLimit, SecRequestBodyLimitAction, SecResponseBodyLimit, SecResponseBodyLimitAction, SecResponseBodyMimeType, SecResponseBodyMimeTypesClear, SecResponseBodyAccess, SecRule, SecRuleInheritance, SecRuleEngine, SecRuleRemoveById, SecRuleRemoveByMsg, SecRuleRemoveByTag, SecRuleScript, SecRuleUpdateActionById, SecRuleUpdateTargetById, SecServerSignature, SecStreamInBodyInspection, SecStreamOutBodyInspection, SecTmpDir, SecUnicodeMapFile, SecUnicodeCodePage, SecUploadDir, SecUploadFileLimit, SecUploadFileMode, SecUploadKeepFiles, SecWebAppId, SecCollectionTimeout
QualysIronBee	REQUEST_HEADER, REQUEST, RESPONSE_HEADER, RESPONSE, POSTPROCESS	AuditEngine, AuditLogBaseDir, AuditLogDirMode, AuditLogFileMode, AuditLogIndex, AuditLogIndexFormat, AuditLogParts, AuditLogSubDirFormat, DefaultBlockStatus, GeoIPDatabaseFile, Hostname, Include, InspectionEngine, LoadModule, Location, Log, LogHandler, LogLevel, ModuleBasePath, PcreMatchLimit, PcreMatchLimitRecursion, PersonalityAdd, PersonalityAliasClear, PersonalityAliasParam, PersonalityClearAll, PersonalityParam, RequestBodyBuffering, RequestBodyBufferLimit, RequestBodyBufferLimitAction, ResponseBodyBuffering, ResponseBodyBufferLimit, ResponseBodyBufferLimitAction, Rule, RuleBasePath, RuleDisable, RuleEnable, RuleEngineLogData, RuleEngineLogLevel, RuleExt, RuleMarker, SensorId, Site, SiteId, StreamInspect
OWASP ESAPI Java-WAF	Before-request-body, after-request-body, before-response[a]	Authentication, autorihization, url, header, virtual-patches, outbond, BeanShell

[a] *Only for Bean-Shell rules.*

Appendix D. Description of extra functions

In this appendix we give a description of extra functions for feature models of SSL/TLS and Fault tolerance.

Feature model	Functions and constraints
SSL/TLS	SSL/TLS → SSL/TLS_ALE = (SSL/TLS_ARO - (SSL/TLS_ARO × (SSL/TLS_AROR)) / 100) × SSL/TLS_SLE
	SSL/TLS_ALE <3.000
	SSL/TLS_SLE = 1000
	SSL/TLS_ARO= 10
	SSL/TLS → SSL/TLS_AROR=(Protocol_AROR+Certificate_AROR+DigitalSignature_AROR)/2
	SSL/TLS.Protocol→ Protocol_AROR = SSLv2.0_AROR + SSLv3.0_AROR + TLSv1.0_AROR + TLSv1.1_AROR + TLS1.2_AROR
	SSL/TLS.Protocol = SSLv2.0 → Protocol_AROR⩾ 20 and Protocol_AROR⩽ 60%
	SSL/TLS.Protocol = SSLv3.0 → Protocol_AROR⩾ 30 and Protocol_AROR⩽ 70%
	SSL/TLS.Protocol = TLSv1 → Protocol_AROR⩾ 40 and Protocol_AROR⩽ 85%
	SSL/TLS.Protocol = TLSv11→ Protocol_AROR⩾ 40 and Protocol_AROR⩽ 90%
	SSL/TLS.Protocol = TLSv12 → Protocol_AROR⩾ 40 and Protocol_AROR⩽ 90%
	SSL/TLS.Certificate→ Certificate_AROR = X509_AROR + OPENPGP_AROR
	¬ SSL/TLS.Certificate→ Certificate_AROR = 0
	SSL/TLS.Certificate.X509 → X509_AROR ⩾ 40 and X509_AROR ⩽ 75%
	¬ SSL/TLS.Certificate.X509 → X509_AROR = 0
	SSL/TLS.Certificate.OPENPGP→ OPENPGP_AROR ⩾ 30 and OPENPGP_AROR ⩽ 70%
	¬ SSL/TLS.Certificate.OPENPGP→ OPENPGP_AROR = 0
	SSL/TLS.DigitalSignature→ DigitalSignature_AROR = SRP_AROR + PSK_AROR + Anonymous_AROR
	SSL/TLS.DigitalSignature.PSK→ PSK_AROR = 0
	SSL/TLS.DigitalSignature.SRP→ SRP_AROR = 0
	SSL/TLS.DigitalSignature.Anonymous→ Anonymous_AROR = 0
	SSL/TLS → SSL/TLS_Cost = Protocol_Cost+ Certificate.Cost + DigitalSignature.Cost
	SSL/TLS.Protocol = SSLv2.0 → Protocol_Cost⩾ 10 and Protocol_Cost⩽ 50
	SSL/TLS.Protocol = SSLv3.0 → Protocol_Cost⩾20 and Protocol_Cost⩽55
	SSL/TLS.Protocol = TLSv1 → Protocol_Cost⩾30 and Protocol_Cost⩽60
	SSL/TLS.Protocol = TLSv11→ Protocol_Cost⩾35 and Protocol_Cost⩽65
	SSL/TLS.Protocol = TLSv12 → Protocol_Cost⩾ 40 and Protocol_Cost⩽70
	SSL/TLS.Certificate→ Certificate_Cost = X509_Cost + OpenPGP_Cost
	SSL/TLS.Certificate.X509 → X509_Cost ⩾ 15 and X509_Cost ⩽ 50
	¬ SSL/TLS.Certificate.X509 → X509_AROR = 0
	SSL/TLS.Certificate.OPENPGP→ OPENPGP_Cost⩾ 25 and OPENPGP_Cost⩽ 40
	¬ SSL/TLS.Certificate.OPENPGP→ OPENPGP_Cost = 0
	SSL/TLS.DigitalSignature→ DigitalSignature_Cost⩾ 10 and DigitalSignature_Cost⩽ 30
Fault tolerance	FT.MTTR = DB.Time + NVP.Time + CP.Time
	FT.MTTR ⩾ 30 and FT.MTTR ⩽ 100
	DB → DB.Time = NoR_Time+Oracle_Time+Binder_Time+BBinder_Time
	DB → DB.NoR_Time⩾ 10 and DB.NoR_Time⩽ 40 and DB.Binder _Time ⩾ 10 and DB.Binder _Time ⩽ 40 and DB.BBinder _Time ⩾ 10 and DB.BBinder _Time ⩽ 40 and DB.Oracle _Time ⩾ 10 and DB.Oracle_Time⩽ 40
	¬ DB→ DB.NoR_Time = 0 and DB.Binder = 0 and DB.BBinder = 0 DB.Oracle_Time = 0
	NVP → NVP.Time = NumberOfVariants.Time + Adjudicator.Time
	NVP → NVP.NoV_Time⩾ 10 and NVP.NoV_Time⩽ 30 and NVP.Adjudicator_Time⩾ 10 and NVP.Adjudicator_Time⩽ 30
	NVP.Adjudicator→ Adjucator_Time=Exact_Time+Median_Time+Mean_Time+Consensus_Time
	¬ NVP → NumberOfVariants.Time = 0 and Adjudicator.Time = 0
	CP → CP_Time = NoS_Time + Oracle_Time +NumberOfCheckPoint_Time
	CP → CP.NoS_Time⩾ 10 and CP.NoS_Time⩽ 40 and CP.Oracle _Time ⩾ 10 and CP.Oracle_Time⩽ 20 and CP.NumberOfCheckPoint_Time⩾ 10 and CP.NumberOfCheckPoint_Time⩽ 30
	¬ CP → CP.NoS_Time = 0and CP.Oracle _Time = 0 CP.NumberOfCheckPoint_Time= 0
	FT → FT.RiskReduction = DB.RiskReduction+NVP.RiskReduction+CP.RiskReduction
	DB → DB.RiskReduction ⩾ 30 and DB.RiskReduction ⩽ 80
	DB.NoR→ DB.NoR_N ⩾ 1 and DB.NoR_N ⩽ 10
	DB.NoR→ DB.NoR_N ⩾1 → DB.Risk Reduction ⩾ 50%
	DB.BinderBackup→ DB.RiskReduction ⩾ 70%
	¬ DB.NoR→ DB.NoR_N = 0
	NVP → NVP. RiskReduction ⩾ 30 and NVP. RiskReduction ⩽ 80%
	NVP.NoV→ (NVP.NoV_N ⩾ 3 and NVP. NoV_N ⩽ 6) → NVP.RiskReduction ⩾ 60%

(*continued on next page*)

Appendix D (*continued*)

Feature model	Functions and constraints
	NVP.NoV→ (NVP.NoV_N $\geqslant$ 7 and NVP.NoV_N $\leqslant$ 9) → NVP.RiskReduction $\geqslant$ 60% NVP.NoV→ (NVP.NoV_N = 2 $*$ x + 1), $x \in$ [1,10] ¬ NVP.NoV→ DB.NoR_N = 0 CP → CP.RiskReduction $\geqslant$ 30 and CP.RiskReduction $\leqslant$ 80% CP.NoS→ CP.NoS_N $\geqslant$ 2 → CP.RiskReduction$\geqslant$ 50% CP.NoS→ CP.NoS_N $\geqslant$ 1 and CP.NoS_N< 2 ¬ CP.NoS→ CP.NoS_N = 0

References

[1] InfoSecurity Europe and PricewaterhouseCoopers, Information Security Breaches Survey 2012. <http://www.pwc.co.uk/en_UK/uk/assets/pdf/olpapp/uk-information-security-breaches-survey-technical-report.pdf>, 2013 (March 2013).
[2] Bonita Soft, Bonita Soft Customer References. <http://www.bonitasoft.com/customers/customer-references>, 2013 (March 2013).
[3] David Kushner, The Real Story of Stuxnet, IEEE Spectrum. <http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet>, 2013 (March 2013).
[4] ENISA, European Network and Information Security Agency. <http://www.enisa.europa.eu/activities/risk-management/current-risk/business-process-integration> (March 2013).
[5] Symantec, Internet Security Threat Report. <http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_2011_21239364.en-us.pdf>, 2013 (March 2013).
[6] SANS, CSIS: 20 Critical Security Controls Version 4.1. <http://www.sans.org/critical-security-controls/>, 2013 (March 2013).
[7] Michael Davis, Research: 2012 Strategic Security Survey, InformationWeek Reports. <http://reports.informationweek.com/abstract/21/8815/security/research-2012-strategic-security-survey.html>, 2012 (March 2013).
[8] Symantec, IT Controls Reference. <www.symantec.com/compliance>, 2009 (March 2013).
[9] A.J. Varela-Vaca, Robert Warschofsky, Rafael M. Gasca, Sergio Pozo, Christoph Meinel, A security pattern-driven approach toward the automation of risk treatment in business processes, in: International Joint Conference CISIS'12-ICEUTÉ 12-SOCÓ 12 Special Sessions, vol. 189, Springer, Berlin Heidelberg, 2013, pp. 13–23.
[10] A.J. Varela-Vaca, Rafael M. Gasca, Sergio Pozo, OPBUS: risk-aware framework for the conformance of security-quality requirements in business processes, in: International Conference on Security and Cryptography (SECRYPT 2011), 2011.
[11] A.J. Varela-Vaca, Rafael M. Gasca, A. Jimenez-Ramirez, A model-driven engineering approach with diagnosis of non-conformance of security objectives in business process models, in: Fifth International Conference on Research Challenges in Information Science (RCIS), pp. 1–6, ISBN:978-1-4244-8670-0, 2011.
[12] A.J. Varela-Vaca, Rafael M. Gasca, Diana Borrego, Sergio Pozo, Fault tolerance framework using model-based diagnosis: towards dependable business processes, International Journal on Advances in Security 4 (1&2) (2011) 11–22. ISSN: 1942–2636.
[13] OPBUS tools. <http://www.lsi.us.es/~quivir/index.php/Main/AJVarelaOPBUS>, 2012 (March 2013).
[14] Wes Sonnenreich, Jason Albanese, Bruce Stout, Return On Security Investment (ROSI): a practical quantitative model, Journal of Research and Practice in Information Technology (2005) 239–252. ISSN: 1443–458X.
[15] E.W. Cope, J.M. Kuster, D. Etzweiler, L.A. Deleris, B. Ray, Incorporating risk into business process models, IBM Journal of Research and Development 54 (3) (2010) 4:1–4:13.
[16] M. zurMuehlen, M. Rosemann, Integrating risks in business process models, in: Proceedings of the Australasian Conference on Information Systems (ACIS), 2005.
[17] J.H. Lambert, R.K. Jennings, N.N. Joshi, Integration of risk identification with business process models, Journal Systems Engineering 9 (3) (2006) 187–198.
[18] L. Churliov, D. Neiger, M. Rosemann, 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.
[19] A. Rodriguez, E. Fernández-Medina, M. Piattini, Towards a UML 2.0 extension for the modeling of security requirements in business processes, Trust and Privacy in Digital Business (2006) 51–61. LNCS, ISBN 978-3-540-37750-4.
[20] M. Menzel, I. Thomas, C. Meinel, Security requirements specification in service-oriented business process management, International Conference on Availability, Reliability and Security (2009) 41–48.
[21] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, C. Meinel, Model-driven business process security requirement specification, Journal of Systems Architecture – Embedded Systems Design 55 (4) (2009) 211–223.
[22] M. Menzel, Model-driven Security in Service-oriented Architectures, Ph.D. Dissertation, Hasso-Plattner Institute, University of Potsdam, 2010.
[23] Y. Asnar, P. Giorgini, Modelling risk and identifying controls in organizations, in: Proceedings of 1st International Workshop on Critical Information Infrastructures Security (CRITIS'06), vol. 4347 of LNCS, Springer, pp. 55–66, 2006.
[24] F. Semmak, C. Gnaho, R. Laleau, L. Maciaszek, C. González-Pérez, S. Jablonski, Extended KAOS method to model variability in requirements, Evaluation of Novel Approaches to Software Engineering, vol. 69, Springer, Berlin Heidelberg, 2010. pp. 193–205 ISBN 978-3-642-14819-4.
[25] D. Mellado, E. Fernández-Medina, M. Piattini, Security requirements engineering framework for software product lines, Information and Software Technology 52 (10) (2010) 1094–1117. ISSN 0950-5849.
[26] Javier Pérez, Miguel A. Laguna, YaniaCrespo González-Carvajal, Bruno González-Baixauli, Requirements variability support through MDA™ and graph transformation, Electronic Notes in Theoretical Computer Science 152 (2006) 161–173. ISSN 1571-066.
[27] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, D. Hughes, Constraint programming as a means to manage configurations in self-adaptive systems, Computer 99 (2012) 1.
[28] Siemens, CRAMM. <http://www.cramm.com/>, 2013 (March 2013).
[29] M-O. Cordier, P. Dague, M. Dumas, F. Lévy, J. Montmain, M. Staroswiecki, L. Travé-Massuyès, A comparative analysis of AI and control theory approaches to model-based diagnosis, in: 14th European Conference on Artificial Intelligence (ECAI), 2000.
[30] J. de Kleer, A.K. Mackworth, R. Reiter, Characterizing diagnoses and systems, Artificial Intelligence 56 (2–3) (1992) 197–222. ISSN 0004-3702.
[31] HangjungZo, Derek L. Nazareth, Hemant K. Jain, Security and performance in service-oriented applications: trading off competing objectives, Decision Support Systems 50 (1) (2010) 336–346.
[32] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990. November.
[33] D. Batory, Feature models, grammars, and propositional formulas, Software Product Lines Conference, Lecture Notes in Computer Sciences, vol. 3714, Springer-Verlag, 2005, pp. 7–20.
[34] D. Benavides, Sergio Segura, Antonio Ruiz-Cortés, Automated analysis of feature models 20 years later: a literature review, Information Systems 35 (6) (2010) 615–636. ISSN 0306-4379.
[35] D. Benavides, A. Ruiz-Cortés, P. Trinidad, Automated reasoning on feature models, in: A d v a nced Information Systems Engineering: 17th International Conference, CAiSE, Lecture Notes in Computer Sciences, vol. 3520, Springer-Verlag, Berlin, 2005, pp. 491–503.
[36] Fidetia, ISO/IEC 27001 Certificate. <http://www.fidetia.es/docs/Cert_27001.pdf> (March 2013).
[37] K. Czarnecki, S. Helsen, U. Eisenecker, Formalizing cardinality-based feature models and their specialization, Software Process: Improvement and Practice 10 (1) (2005) 7–29. ISSN: 1099-1670.
[38] A. Avizienis, J.-C. Laprie, B. Randell, C.E. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transaction on Dependable and Secure Computing 1 (1) (2004) 11–33.
[39] G. Dobson, Using WS-BPEL to implement software fault tolerance for web services, in: SEAA '06, 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, 2006, pp. 126–133.
[40] V.O. Onditi, G. Dobson, J. Hutchinson, J. Walkerdine, P. Sawyer, Specifying and constructing a fault-tolerant composite service, 2008. ECOWS '08, IEEE Sixth European Conference on Web Services, 2008, pp. 135–142.
[41] Kipp Hickman, The SSL Protocol, Netscape Communications Corp., 1995.
[42] T. Dierks, C. Allen, The TLS Protocol Version 1.0, RFC 2246, January 1999.
[43] Oracle Java™ Secure Socket Extension (JSSE). <http://docs.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html#Features>, 2012 (March 2013).
[44] OpenSSL. <http://www.openssl.org/>, 2012 (March 2013).
[45] GnuTLS. <http://www.gnu.org/software/gnutls/>, 2012 (March 2013).
[46] R. Macfarlane, W. Buchanan, E. Ekonomou, O. Uthmani, L. Fan, O. Lo, Review of security policy implementations, Computers & Security (2012). ISSN: 0167-4048.

[47] N. Gupta, A. Saikia, Web Application Firewall, Tech. Report: CS499, Department of Computer Science and Engineering Indian Institute of Technology, Kanpur, 2007.
[48] Maximilian Dermann, MirkoDziadzka Boris Hemkemeier, Achim Hoffmann, Alexander Meisel, Matthias Rohr, Thomas Schreiber, OWASP – Best Practice: Use of Web application Firewalls, 2008.
[49] TrustwaveModSecurity. <http://www.modsecurity.org/>, 2012 (March 2013).
[50] QualysIronbee. <http://www.ironbee.com/>, 2012 (March 2013).
[51] OWASP ESAPI WAF. <https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API>, 2012 (March 2013).
[52] AQTronixWebKnight. <http://www.aqtronix.com>, 2012 (March 2013).
[53] Stingray Application Firewall. <http://www.riverbed.com/>, 2012 (March 2013).
[54] WebCastellum. <http://www.webcastellum.org>, 2012 (March 2013).
[55] Zion Secured. <http://www.zionsecurity.com/products/zion-secured>, 2012 (March 2013).
[56] Guardian@JUMPERZ.NET. <http://guardian.jumperz.net>, 2012 (March 2013).
[57] EasyWAF. <http://www.easywaf.com>, 2012 (March 2013).
[58] Dynadec Decision Technologies, COMET®. <http://dynadec.com>, 2012 (March 2013).
[59] D. Benavides, S. Segura, P. Trinidad, A. Ruiz-Cortés, FAMA: tooling a framework for the automated analysis of feature models, in: First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS), 2007.
[60] R. Mazo, C. Salinesi, D. Diaz, VariaMos: a tool for product line driven systems engineering with a constraint based approach, in: 24th International Conference on Advanced Information Systems Engineering (CAiSE Forum'12), 2012, pp. 1–8.
[61] Pure::system. <http://www.pure-systems.com/>, 2012 (March 2013).
[62] T. Neubauer, J. Heurix, Defining secure business processes with respect to multiple objectives, in: Proceedings of International Conference Availability, Reliability and Security, Third International Conference on Availability, Reliability and Security (ARES), 2008, pp. 187–194.
[63] ISO27001: Information Security Management System (ISMS) standard. <http://www.27000.org/iso-27001.htm>, 2005 (March 2013).
[64] T. Neubauer, A. Ekelhart, S. Fenz, Interactive selection of ISO 27001 controls under multiple objectives, in: Proceedings of The IFIP Tc 11 23rd International Information Security Conference, Springer, 2008, pp. 477–492 ISBN: 978-0-387-09698-8.
[65] M. Ehrgott, X. Gandibleux, Multiobjective combinatorial optimization – theory, methodology, and applications, in: Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, vol. 52 of International Series in Operations Research & Management Science, 2003, pp. 369–444.
[66] R. Barabanov, S. Kowalski, L. Yngström, Information Security Metrics: State of the Art: State of the art, DSV Report Series No. 11-007. <su.diva-portal.org/smash/get/diva2:469570/FULLTEXT01>, 2011.
[67] MAGERIT 2. <http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=PAE_PG_CTT_General&langPae=es&iniciativa=184>, 2012 (March 2013).
[68] T. Dierks, E. Rescolar, The Transport Layer Security (TLS) Protocol Version 1.1, RFC 4346, April 2006.
[69] T. Dierks, E. Rescolar, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, August 2008.
[70] A. Freier, P. Karlton, P. Kocher, The Secure Sockets Layer (SSL) Protocol Version 3.0, RFC 6101, August 2011.
[71] William Stallings, SSL: Foundation for Web Security, The Internet Protocol Journal – vol. 1(1), CISCO. <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html>, 2012 (March 2013).