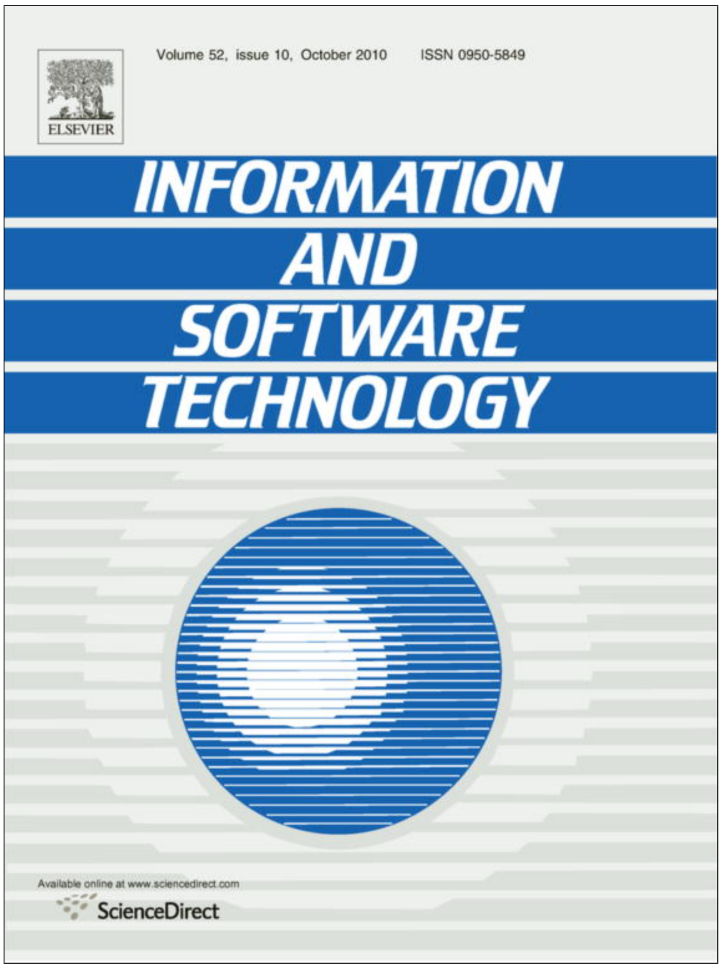

Volume 52, issue 10, October 2010
ISSN 0950-5849
ELSEVIER
INFORMATION AND SOFTWARE TECHNOLOGY
Available online at www.sciencedirect.com
ScienceDirect

Information and Software Technology 52 (2010) 1094–1117

Contents lists available at ScienceDirect

Information and Software Technology

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

Security requirements engineering framework for software product lines

Daniel Mellado [a], Eduardo Fernández-Medina [b,*], Mario Piattini [b]

[a] Spanish Tax Agency, Large Taxpayers Department – IT Audit Unit Paseo de la Castellana 106, 28046 Madrid, Spain
[b] University of Castilla-La Mancha. Alarcos Research Group, Information Systems and Technologies Institute, Information Systems and Technologies Department, ESI, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 16 October 2009
Received in revised form 3 May 2010
Accepted 11 May 2010
Available online 31 May 2010

Keywords:
Security requirements engineering
Security software engineering
Product lines
Requirements engineering
Security requirement
ISO 27001

ABSTRACT

Context: The correct analysis and understanding of security requirements are important because they assist in the discovery of any security or requirement defects or mistakes during the early stages of development. Security requirements engineering is therefore both a central task and a critical success factor in product line development owing to the complexity and extensive nature of software product lines (SPL). However, most of the current SPL practices in requirements engineering do not adequately address security requirements engineering.
Objective: The aim of this approach is to describe a holistic security requirements engineering framework with which to facilitate the development of secure SPLs and their derived products. It will conform with the most relevant security standards with regard to the management of security requirements, such as ISO/IEC 27001 and ISO/IEC 15408.
Results: This framework is composed of: a security requirements engineering process for SPL (SREPPLine) driven by security standards; a Security Reference Meta Model to manage the variability of those SPL artefacts related to security requirements; and a tool (SREPPLineTool) which implements the meta-model and supports the process.
Method: A complete explanation of the framework will be provided. The process will be formally specified with SPEM 2.0 and the repository will be formally specified with an XML grammar. The application of SREPPLine and SREPPLineTool will be illustrated through a description of a simple example as a preliminary validation.
Conclusion: Although there have been several attempts to fill the gap between requirements engineering and SPL requirements engineering, no systematic approach with which to define security quality requirements and to manage their variability and their related security artefacts in SPL models is, as yet, available. The contribution of this work is that of providing a systematic approach for the management of the security requirements and their variability from the early stages of product line development in order to facilitate the conformance of SPL products with the most relevant security standards.

1. Introduction

In recent years we have observed that more and more organizations have found themselves in difficulties as a result of security breaches. In fact, according to the statistics provided by the Software Engineering Institute's CERT Coordination Centre, the number of reported application vulnerabilities rose from 171 in 1995 to 7236 in 2007 and to 6058 in the first nine months of 2008 [12]. The tendency towards larger systems that are distributed throughout the Internet has caused many new security threats [52], since modern web applications are no longer mere hypertextual information repositories, but are complex distributed systems [5]. This signifies that present-day information systems are vulnerable to a host of threats and cyber-attackers such as malicious hackers, code writers and cyber-terrorists [15].

In recent years, many public and private organizations have altered their way of thinking about their business processes in order to improve the quality of delivered services by achieving better efficiency and efficacy [4]. There has been a simultaneous increase in both the demand for and the complexity of the software needed. The need to obtain both high-quality information systems and higher productivity has led software product line (SPL) based development to become the most successful approach in the reuse field, since it can assist in significantly reducing both time-to-market and development costs [10,13]. The SPL development paradigm is based on increasing the reuse of all types of artefacts, thanks to the combination of coarse-grained components with a top-down systematic approach in which software components are integrated into a high-level structure.

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: damefe@esdebian.org (D. Mellado), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina), Mario.Piattini@uclm.es (M. Piattini).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.05.007

The complexity and extensive nature of product line development signifies that security and requirements engineering are critical success factors in the development of a software product line. It should consequently be subject to careful requirements analysis and decision making, given that a weakness in security may cause problems in all the products of a product line. Many requirements engineering practices must also be appropriately tailored to the specific demands of product lines [9]. The specification of requirements for an SPL is therefore a challenging task [48]. The specification of security quality requirements for an SPL is even more challenging as a result of the varying security properties required in different products for the diversity of market segments and the constraint of simultaneously maintaining the cost-effective principle of the SPL paradigm.

The principle which establishes that the building of security during the early stages of the development process is cost-effective and also brings about more robust designs is also widely accepted [31]. This has meant that the discipline of Security Requirements Engineering is now considered to be an extremely important part of the SPL development process for the achievement of secure SPL and products. It provides techniques, methods, standards and systematic and repeatable procedures for tackling SPL security requirement issues throughout the SPL development lifecycle, both to ensure the definition of security quality requirements and to manage the variability of security properties.

Nevertheless, software engineering methodologies and the standard proposals of SPL engineering have traditionally ignored security requirements and security variability issues. The few recent proposals which deal with security in SPLs, such as [3,16], focus mainly on the design of implementation aspects of SPL development or include only a few security requirements activities. After analysing the most relevant current "generic" security requirements related proposals in [44], we can conclude that none of them are either sufficiently specific or are tailored to the SPL development paradigm.

In this paper we propose a security requirements engineering framework for SPL which is an evolution of our previous "generic" security requirements engineering process (SREP) [42],. We will provide a complete explanation of the Security Requirements Engineering Process for Software Product Lines (SREPPLine). We will also described the activities of which SREPPLine is composed, which are formally specified with SPEM 2.0 [51] (an OMG standard). An explanation will also be provided of the Security Reference Meta Model, which assists in the management of the variability of SPLs. A simple example will be described, which will focus on security requirements artefacts for an e-billing reception service product line for Spanish public administrations. This will serve to illustrate the application of SREPPLine, the Security Reference Meta Model and SREPPLineTool (the prototype tool developed to provide automated support to SREPPLine).

In previous works we have only outlined SREPPLine and SREPPLineTool respectively, along with a case study as a preliminary validation of the application of the domain activity of SREPPLine (PLSecDomReq) [45,47]. The SPL security variability management was presented and briefly detailed in another previous work [43], but without any explanation of the integration of the models proposed and without the inclusion of an XML grammar. This paper, in contrast with our previous works, will therefore provide a detailed explanation, and the SREPPLine process will be formally specified with the SPEM 2.0 standard [51], specifying: roles, input and output artefacts, activities, tasks and steps. A definition of a grammar in XML for the Security Reference Meta Model will also be given. By using these standards, SREPPLine better facilitates integration with other processes and repositories. We shall also present a framework in which these components will be integrated with the aim of providing a security requirements framework for SPL. This framework will systematically deal with the security requirements artefacts and their variability from the early stages of SPL development and its products. This will facilitate SPL products' conformance with the most relevant security standards with regard to the management of security requirements, such as ISO/IEC 27001 [26] and ISO/IEC 15408:2005 [25] (Common Criteria).

The remainder of the paper is organized as follows: Section 2 presents related work in security requirements engineering for SPLs. In Section 3, we shall describe our proposed security requirements engineering framework for SPLs, along with an explanation of the Security Reference Meta Model and SREPPLine process activities, which will be formally specified with SPEM 2.0 [51]. In Section 4, we will show a simple example of the SREPPLine application in order to provide a preliminary validation. Finally, in Section 5, we shall discuss our contributions and future work.

2. Related work

There has, in recent years, been a spectacular growth in the number of security standard and security requirements related proposals. Several attempts have also recently been made to define SPL architectures for security. We shall now outline those proposals which are particularly close in subject matter to ours and which will then be compared to SREPPLine.

The "SPL reference architecture for security" approach of Faegri and Hallsteinsen [16] suggests a reference architecture which draws upon state-of-the-art techniques and practices from SPL engineering and information security, and constitutes a decision support framework for security architecture design in SPLs. The authors propose a structured repository to support architectural design with which to capture and manage knowledge related to this.

The "Architecture reasoning for supporting SPL evolution" approach of Arciniegas et al. [3] proposes a new process to support SPL evolution which involves architecture recovery and conformance methods and a set of techniques and tools to support them. The authors also present a case study dealing with non-functional security requirements in distributed environments through which to create a reference architecture. They also propose ways in which to enhance the coverage of architectural security requirements.

In contrast to SREPPLine, the existing proposals with regard to SPL security, despite tackling security management in SPL engineering, are more orientated towards the software solution than to security requirements. The most relevant current "generic" security requirements related proposals are those such as: security use cases [17,19]; misuse cases [52,58,59]; UMLsec [6,27–29]; a framework for security requirements engineering [22]; SQUARE [1,39,40]; security requirements methods based on i^* framework [36,62]; Secure Tropos [14,20,38]; security requirements based on intentional anti-models [33]; and SIREN [34,61]. These proposals are neither sufficiently specific nor are they tailored to the SPL development paradigm, principally because they do not deal with security requirements variability, which is an essential aspect of the SPL development paradigm. Moreover, they do not provide a methodological tailored approach for SPL engineering.

In contrast to SREPPLine, none of the proposals provides a systematic approach through which to manage and trace security requirements with the aim of including security requirements variability in SPL models. Furthermore, these "generic" security requirements engineering approaches do not consider certain specific tasks and techniques which are important for the SPL development paradigm, such as security feature identification or security feature variability, i.e., security variation points and variant identification. SREPPLine, on the other hand, takes these tasks and techniques into account, integrates them into the repository, specifies them in activities which are composed of tasks and defines them in steps and guidelines (standards, techniques and practices). These approaches do not manage the security requirements variability management required for carrying out an instantiation of a new product in the SPL. SREPPLine, however, uses the Security Reference Model implemented by a security repository to facilitate the traceability management of all the security artefacts involved in a product instantiation of the SPL (security features, assets, security objectives, threats, risks and security requirements, along with their respective variability relations). This signifies that it possible to trace all the affected security requirements artefacts for a given variant or variation point or vice-versa. This Security Reference Model could also be used to link SREPPLine artefacts to the software artefacts specified in other models, i.e., feature models, use cases or misuse cases. Our proposal therefore provides security variability consistency across the different SPL security artefacts.

After analyzing most of the aforementioned proposals in [41,46], we also concluded that none of them facilitates the SPL products security certification against the most relevant international security standards with regard to the management of security requirements, such as, principally, ISO/IEC 15408 [25], ISO/IEC 27001 [26]. Neither do these security standards alone provide a methodology for developing either secure systems or SPLs. There was thus a gap in these issues (security requirements, security standards, variability and SPL) that our proposal (SREPPLine) attempts to fill. Having said this, each of these proposals makes highly important contributions to security requirements engineering. In addition, some of their features are used as the basis for SREPPLine.

3. Security requirements engineering framework for software product lines

This section shows the main concepts of requirements engineering in SPLs in order to provide a better comprehension of our approach. The key components of the security requirements engineering framework for SPLs are also shown (see Fig. 1). We shall therefore summarize the most important characteristics of our proposed process, SREPPLine, along with an outline of the prototype tool we have developed to support it.

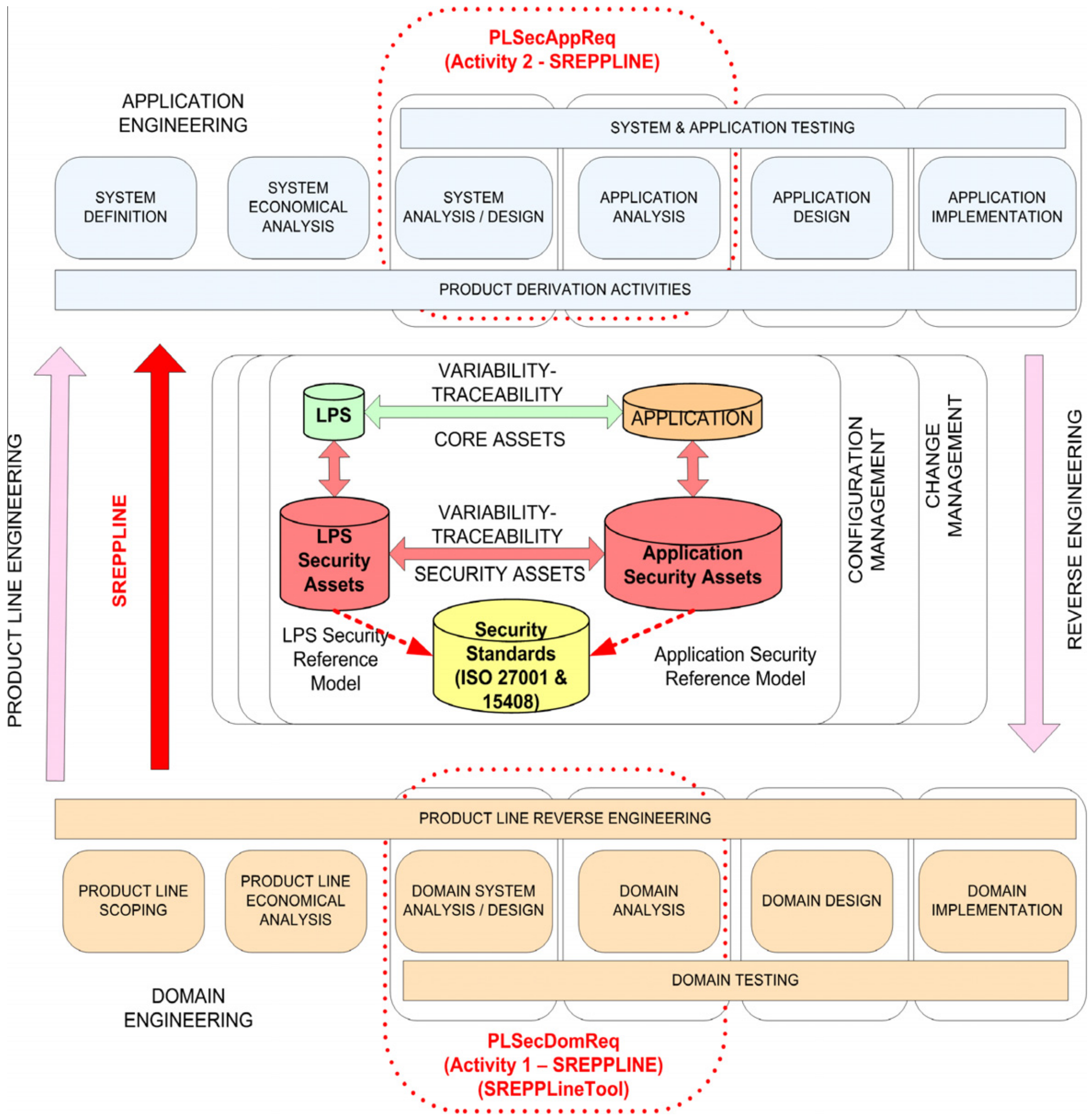


Fig. 1. Security requirements engineering framework for software product lines (SPLs or LPS).

3.1. Software product line requirements engineering basics

A Software product line (SPL) is a set of software-intensive systems sharing a common, managed set of features [30] which satisfy the specific needs of a particular market segment or mission and which is developed from a common set of core assets in a prescribed way [13]. Exploiting commonalities between different systems is at the heart of Software Product Line Engineering. These commonalities and differences are described by using the core concept in Software Product Line Engineering: variability. Variability describes the variations in both functional and non-functional features in the product line. Features are either a commonality or a variation. Variability management is the activity in product line development that aims to model a product line as a whole and to customize or change specific product line members. Its importance signifies that it can actually be seen as the key feature that distinguishes product line development from other approaches to software development [55]. In common language use the term variability refers to the ability or the tendency to change, but in this case this change does not occur by chance but is brought about deliberately. For example: an electric bulb can be lit or unlit, or a software application can support different languages. Variability in SPL is therefore variability that is modelled to enable the development of customized applications by reusing predefined, adjustable artefacts. The variability of a SPL thus distinguishes different applications of the product line. In contrast to variability, the commonality in SPL denotes features that are part of each application in exactly the same form. This means that it is often possible to de-

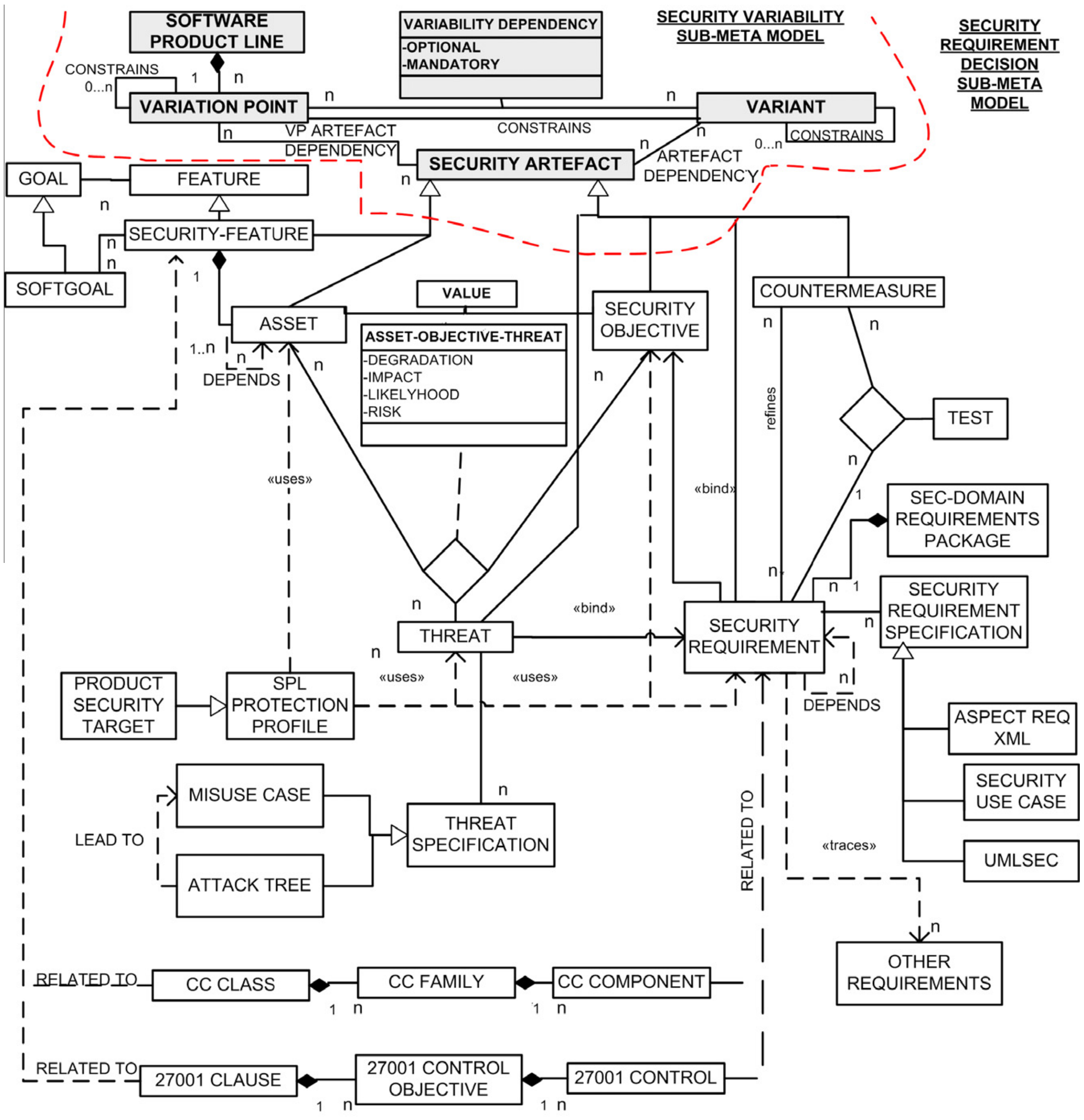


Fig. 2. Security Reference Meta Model.

cide whether a feature is a variable of the SPL or whether it is common to all software product applications, and thus adds to the commonality.

The Software Product Line Engineering paradigm differentiates two processes: domain engineering and application engineering [53]. Domain engineering is the process of SPL engineering in which commonality and variability of the product line are defined and carried out. According to [53] the domain requirements engineering sub-process encompasses all activities for eliciting and documenting the common and variable requirements of the product line. Application engineering is the process of SPL engineering in which the applications of the product line are built by reusing domain artefacts and exploiting product line variability. Product line requirements define the products and their common and variable features in the product line. Requirements that are common to the entire family, which constitute the product line requirements and an important core asset, should be managed separately from requirements that are particular to a subset of the products (or to a single product), which must also be managed. The SPL scope binds the products included in the product line: product line requirements refine the scope by more precisely defining the characteristics of the products in the product line. Both concepts are closely coupled and evolve together [13].

3.2. Overview of our approach

Security Requirements Engineering Process for Software Product Lines (SREPPLine) is an iterative and incremental process which is an add-in of tasks that can be incorporated into and tailored to an organization's SPL development process model to provide it with a security requirements engineering approach. We have defined the key tasks that must be part of each SPL activity, signifying that the order in which the steps are performed depends on the particular process that is established in an organization. The activities and their tasks can thus be combined with existing development methods such as RUP (Rational Unified Process), OpenUP (Open Unified Process), or other development processes, although in this paper we describe integration with the framework proposed for SPL engineering by Bühne et al. in [11]. It can therefore be termed as a scalable process since not all the tasks and steps are required, and developers could create their own lightweight process by selecting a subset of the steps in each task.

It is a security-feature or security-goal based process which is driven by risk and security standards (specifically ISO/IEC 27001 and Common Criteria). It deals with security requirements and their related artefacts from the early phases of SPL development in a systematic and intuitive manner especially tailored to SPL based development. It is based on the use of the latest and most widely validated security requirements techniques, such as security use cases [17] or misuse cases [59], along with the integration of the Common Criteria (CC) components and ISO/IEC 27001 controls into the SPL lifecycle in order to facilitate SPL product security certification (depicted as cylinders in the centre of Fig. 1). Our proposed process suggests the use of a method to carry out risk assessment which conforms to ISO/IEC 13335 [23]. It specifically uses Magerit [37], the methodology officially recognised by NATO at the 9th NATO cyberdefense workshop in 2008 and by OECD [49], for both SPL risk assessment and SPL products risk assessment. The aim of SREPPLine is to minimize both knowledge of the necessary security standards and security expert participation during SPL product development. To this end, it provides a Security Reference Model (shown in Fig. 1) to facilitate security artefact reuse and to implement the Security Reference Meta Model. This meta-model is composed of the Security Variability Sub-Meta Model and the Security Requirement Decision Sub-Meta Model, both of which assist in the management of the variability and traceability of the security requirements related artefacts of the SPL and its products. The meta-model is the basis used by the SREPPLine tasks to capture, represent and share knowledge about security requirements for SPL and help to certify them against security standards. In essence, it is a knowledge repository with a structure to support security requirements reasoning in SPL.

Fig. 1 shows the typical activities of both application engineering (at the top of the figure) and domain engineering (at the bottom of the figure) based on the framework for SPL engineering proposed by Bühne et al. in [11] (shown in rounded squares in Fig. 1). The integration of the Product Line Security Domain Requirements Engineering (PLSecDomReq) and Product Line Security Application Requirements Engineering (PLSecAppReq) SREPPLine activities within common activities of domain engineering and application engineering is also described, and this integration is depicted in the figure with a dotted line. That is, SREPPLine activities should be integrated into the analysis activities of the domain and application engineering processes of an organization whose software development paradigm is based on SPL engineering. In the centre of the figure there is an arrow denominated as SREPPLine which indicates the direction of our process. This direction is the same as that of Product Line Engineering, since our proposed process must be integrated into SPL engineering. It is not designed to be integrated into a reverse engineering process. The repositories that manage the traceability and variability and implement the Security Reference Model are also represented in the centre of the figure, in which the repositories are depicted as three-dimensional ellipses. The links between the SPL repositories and the repositories of its derived applications are also shown. There are at least five types of necessary repositories:

- *LPS repository*: This maintains the common artefacts of the product line and the links with the artefacts of their derived applications.
- *Application repository:* There is one of these types of repository for each application derived from the Product Line. This repository registers the artefacts derived from the product line and the application's specific artefacts.
- *LPS Security Assets repository:* This manages the product line security artefacts (assets, security objectives, threats, security requirements, etc.) and the relations with the security artefacts of their derived applications, the security standards and the related artefacts of the product line (such as, for example, accessibility requirements).
- *Application Security Assets repository:* There is one of these types of repository for each application derived from the Product Line. This repository manages the security artefacts derived from the product line and the specific security artefacts of the applica-

```
Activity {kind = Phase}: Name of the Phase
    Process: SREPPLine

    Activity {kind = Iteration}: Name of the iterative Activity
        TaskUse: Name of the task
            ProcessPerformer {kind: primary}
                RoleUse: Role Name {kind: in}
            WorkDefinitionParameter {kind: in}
                WorkProductUse : Artefact name
            WorkDefinitionParameter {kind: out}
                WorkProductUse : Artefact name {state: state}
            Steps
                Step: Name of the step
            Guidance
                Guidance {kind: type}: Name
```

Fig. 3. SREPPLine structure using SPEM 2.0.

tion. It also manages the links with the related artefacts of the application and the security standards elements.

- The Security Standards repository registers the elements of the security standards (such as ISO/IEC 27001 controls or ISO/IEC 15408 components).

Finally, the management of these repositories is performed by the prototype tool that we have developed to provide automated support to SREPPLine, SREPPLineTool. This prototype implements the Security Reference Meta Model by means of dynamic repositories of security artefacts, and guides us in the execution of the process in a sequential manner. This tool thus permits us to apply the SREPPLine process in an SPL development by providing automated support to its activities.

3.3. Security Reference Meta Model

The existing variability management approaches, such as, for example: [60,54,7] are focused on addressing functional requirements variability. In contrast, in our proposed security requirements engineering framework for SPL we suggest using the Security Reference Meta Model which is focused on managing the security requirements artefacts' variability elicitation, management and representation. As Fig. 2 show, our proposed meta-model is composed of two Sub-Meta Models: the Security Variability Sub-Meta Model and the Security Requirement Decision Sub-Meta Model. These are complementary to each other and are represented by using UML 2.0. The meta-model is based on the Reusable Assets Specification (RAS), adopted as an OMG standard [50] and

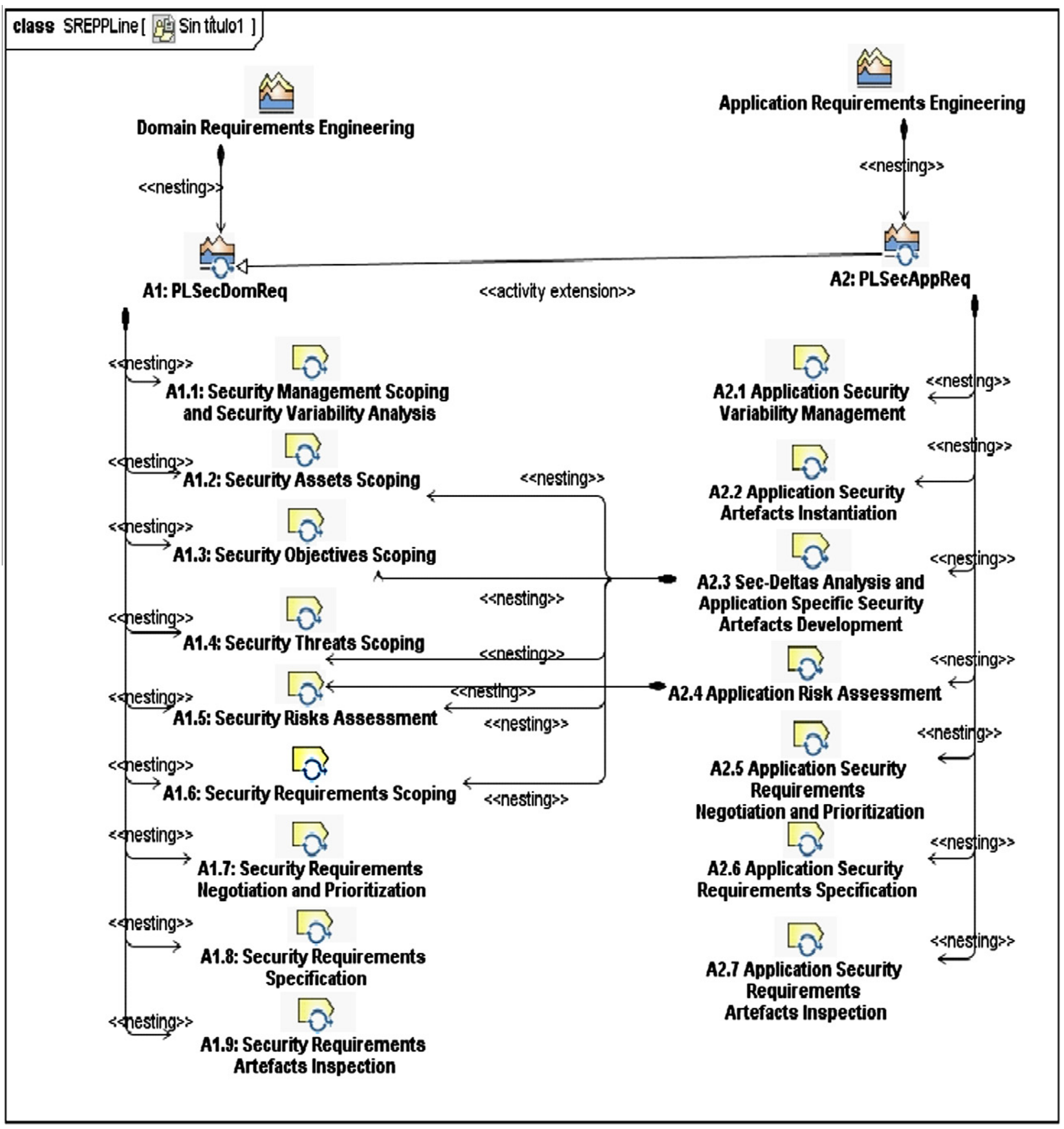

class SREPPLine [Sin título1]
Domain Requirements Engineering
Application Requirements Engineering
<<nesting>>
A1: PLSecDomReq
<<activity extension>>
A2: PLSecAppReq
A1.1: Security Management Scoping and Security Variability Analysis
A1.2: Security Assets Scoping
A1.3: Security Objectives Scoping
A1.4: Security Threats Scoping
A1.5: Security Risks Assessment
A1.6: Security Requirements Scoping
A1.7: Security Requirements Negotiation and Prioritization
A1.8: Security Requirements Specification
A1.9: Security Requirements Artefacts Inspection
A2.1 Application Security Variability Management
A2.2 Application Security Artefacts Instantiation
A2.3 Sec-Deltas Analysis and Application Specific Security Artefacts Development
A2.4 Application Risk Assessment
A2.5 Application Security Requirements Negotiation and Prioritization
A2.6 Application Security Requirements Specification
A2.7 Application Security Requirements Artefacts Inspection

Fig. 4. SREPPLine activities structure using SPEM 2.0.

Table 1
SREPPLine's most important work products.

Security features	Those features (concept of feature taken from Kang et al. [30]) which describe security aspects of the system (application or product line)
Asset	Anything that has value to the organization (ISO/IEC 13335) [23]
Security objective	The objectives which must be achieved in order to protect the organization's business goals
Threat	A potential cause of an unwanted incident, which may result in harm to a system or organization (ISO/IEC 13335) [23]
Risk	An estimate of the degree of exposure to threat to one or more assets causing damage or prejudice to the organization (Magerit [37])
ISO/IEC 27001	ISO/IEC standard 27001 Information technology – Security techniques – Information security management systems – Requirements. SREPPLine integrates and uses ISO/IEC 27001 clauses, control objectives and controls
ISO/IEC 15408	ISO/IEC standard 15408 2005 Information technology – Security techniques – Evaluation criteria for IT security. (Common Criteria v.3). SREPPLine integrates and uses Common Criteria classes, families and components
Protection Profile	(Compatible with ISO/IEC 15408 definition) an implementation-independent statement of security needs for a Product Line
Security Target	(Compatible with ISO/IEC 15408 definition) an implementation-dependent statement of security needs for a specific identified product
IEEE 830	IEEE 830 1998 Recommended Practice for Software Requirements Specifications. 1998
Sec-delta	Sec-deltas occur when stakeholder security requirements cannot be completely satisfied by security domain requirements artefacts

Table 2
SREPPLine most important guidance.

Feature matrix [30]	A technique to identify common and variable features between products
Attack trees [52,56]	Conceptual diagrams of threats to systems and possible attacks to reach these threats
UMLSec [27]	A UML extension to specify requirements regarding confidentiality and integrity in analysis models in order to develop secure systems
Magerit techniques [37]	Magerit is an open methodology for Risk Analysis and Management, developed by the Spanish Ministry of Public Administrations and offers a framework and guideline to Public Administration. Given its open nature it is also used outside the Administration. It has been structured into three books, one of which contains the methodology, one with a catalogue of elements; and one containing practical techniques, which describes techniques frequently used to carry out risk analysis
Delphi evaluation [35]	A systematic, interactive forecasting method which relies on a panel of experts who anonymously answer questionnaires in rounds
EasyWinWin negotiation [21]	Both a collaborative platform and a step-by-step methodology for bringing key stakeholders together to hammer out a set of requirements with which they can all live
Misuse case [59]	A business process modelling tool which describes the steps involved in performing a malicious act against a system, just as an act that the system is supposed to perform in a use case would be described
Security use case [18]	A specialized use case, the scope of which is restricted to security issues
+SAFE [57]	An extension to CMMI-DEV that covers security and safety engineering

also extends the orthogonal variability meta-model of Pohl et al. proposed in [53].

We have also defined an XML grammar in order to facilitate the formalization and standardization of the Security Reference Model. The main objective of using XML is to facilitate integration with other repositories and/or tools which might exist in an organization, along with facilitating the extension of our proposed repository by an organization in which SREPPLine has been adopted. The core of this XML grammar is shown in Appendix A (from Figs. 8–12).

Table 3
SREPPLine roles.

Client	Organization that requests a system, product, service or product line
Expert users	Final users who will use the product for their work and who have a good knowledge of the business in which the product will be used
Requirements engineer	This role is in charge of the task of capturing, structuring, and accurately representing the user's requirements so that they can be correctly embodied in systems which meet these requirements
Business domain experts (only in PLSecDomReq)	The business experts have a profound knowledge of the product line domain and are thus able to outline both the features of the products of the domain and the product line business goals and features
PL manager	This role is in charge of managing the product line and is responsible for product line final decisions as regards cross-cutting issues, such as commonalities and variabilites of the product line, the realisation of new versions or new product instantiation
Security requirements engineer	This is the key role and it participates in and leads most of the activities
Security expert	This is the specialized role in security and the main task of this role is to improve the overall security of the development process
Security architect	The role of the security architect is to design the technical architecture that will later be implemented in the process
Inspection team	This role is responsible for applying the principles and practices of software quality assurance throughout the software development life cycle

Table 4
Activity 1 (PLSecDomReq) definition.

Activity {kind = Phase}: **Domain Requirements Engineering**
Process: **SREPPLine**

Activity {kind = Iteration}: First **Product Line Security Domain Requirements Engineering (PLSecDomReq)**

As was previously explained, the Security Reference Model is composed of two Sub-Meta Models. The Security Variability Sub-Meta Model is used to assist in the management and representation of variability and traceability of the security requirements related artefacts of the SPL and its products, along with the SPL and its products security standards certification. The Security Requirement Decision Sub-Meta Model supports the elicitation of security artefacts variability, in addition to capturing, specifying and reasoning about security requirements and their artefacts for the SPL members. It also supports the development of a security requirement protection profile[1] for the security features of the SPL, along with the development of a security target[2] for the security features of a product of the SPL. It is additionally helpful in the pro-

[1] Protection profile (PP) (compatible with ISO/IEC 15408 definition): an implementation-independent statement of security needs for a Product Line.
[2] Security target (ST) (compatible with ISO/IEC 15408 definition): an implementation-dependent statement of security needs for a specific identified product.

Table 5
Task A1.1 definition.

TaskUse: A1.1 – **Security Management Scoping and Security Variability Analysis**
ProcessPerformer {kind: **primary**}

RoleUse: **Product line manager** {kind: in}
RoleUse: **Business domain experts** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Stakeholder needs**
WorkProductUse: **Existing products of the domain**
WorkProductUse: **Business goals**
WorkProductUse: **Features and feature model**
WorkProductUse: **Organization security policy**
WorkProductUse: **Law and regulations**
WorkProductUse: **Security standards (ISO/IEC 27001 clauses and ISO/IEC 15408 classes)**
WorkProductUse: **Requests for additional/altered security features**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable security features** {state: **initial draft**}
WorkProductUse: **List of similar Protection Profiles** {state: initial draft}
WorkProductUse: **Security variability sub-model** {state: **initial draft**}
WorkProductUse: **Security requirement decision sub-model** {state: **initial draft**}

Steps
Step: A1.1.1 **Analyze the requests for additional/altered security features**
Step: A1.1.2 **Define security goals based on stakeholders needs, business goals, security policy and regulations**
Step: A1.1.3 **Determine the type of category of the product line**
Step: A1.1.4 **Determine the relevant ISO/IEC 27001 clauses and ISO/IEC 15408 classes**
Step: A1.1.5 **Identify similar product line Protection Profile**
Step: A1.1.6 **Select the security features from product line existing features**
Step: A1.1.7 **Identify new security features**
Step: A1.1.8 **Analyze the security features commonality**
Step: A1.1.9 **Analyze the security features variability**
Step: A1.1.10 **Relate each security variable artefacts (security features and security standards) to the corresponding variants or variation points in the security reference model**
Step: A1.1.11 **Model security variability**
Step: A1.1.12 **Inspect security features** & **variability model**

Guidance
Guidance {kind: **Practice**}**: Questionnaire**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Application features matrix**
Guidance {kind: **Practice**}**: Hierarchical security features**
Guidance {kind: **Checklist**}**: Organization policy, laws and standards**

Table 6
Task A1.2 definition.

TaskUse: A1.2 **Security Assets Scoping**
ProcessPerformer {kind: primary}

RoleUse: **Product line manager** {kind: in}
RoleUse: **Business domain experts** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Family assets**
WorkProductUse: **Common and variable security features**
WorkProductUse: **Security reference model**
WorkProductUse: **Organization inventories**
WorkProductUse: **Business processes**
WorkProductUse: **Scenarios**
WorkProductUse: **Personal data law**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable security assets** {state: **initial draft**}
WorkProductUse: Graph **of assets dependences** {state: **initial draft**}
WorkProductUse: **Security requirement decision model** {state: **initial draft**}

Steps
Step: A1.2.1 **Identify and select security assets**
Step: A1.2.2 **Identify common and variable security assets**
Step: A1.2.3 **Determine security assumptions**
Step: A1.2.4 **Categorize security assets**
Step: A1.2.5 **Identify security assets dependencies**
Step: A1.2.6 **Represent security assets variability and dependencies**
Step: A1.2.7 **Inspect security assets graph** & **variability model**

Guidance
Guidance {kind: **Checklist**}**: Catalogue of type of security assets according to Magerit**
Guidance {kind: **Practice**}**: Attack trees**
Guidance {kind: **Practice**}**: Graph of dependencies of assets**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Application requirements matrix**

cess of determining the most appropriate security requirements artefacts and security standards.

This Variability Sub-Meta Model relates the defined variability to other software development models such as feature models, use case or misuse case or security use case models, design models and test models. It thus provides a cross-cutting view of security requirements variability in all security development artefacts and assists in maintaining the different views of variable security requirements artefacts consistent.

In Fig. 2, we have used the elements *variant* and *variation point* to manage the variability of SPL engineering. A *variation point* (VP) represents a variation subject and defines what can vary in an SPL (“*what varies?*”). A *variant* (V) represents the variation object and defines a concrete type of variation (“*how does it vary?*”). A variant can be related to one or more variation points (multiplicity $1 \cdots n$) and a variation point associates at least one variability dependency to one variant (multiplicity $1 \cdots n$). A variant can also *constrain* (<<require>> or <<exclude>>) variants and a variant can be constrained by several variants (multiplicity $0 \cdots n$ at both ends). For example, in Fig. 5, the variant ‘Internet web’ for an e-billing reception system of a Spanish Public Administration ‘requires’ the variant ‘https’ (owing to a legal requirement of the case study). In order to relate the variability defined in the Variability Sub-Meta Model to the software artefacts specified in other models, this meta-model contains the element *security artefact*, which is a specialization of an artefact. The assignation of (an) *artefact(s)* to a *variation point* makes it possible to represent the common behaviour of several *variants*. However, an artefact can but does not have to be related to one or several variation points and vice-versa (multiplicity $0 \cdots n$ both sides). By assigning (an) *artefact(s)* to a variant, the *artefact* becomes variable. The *artefact* which is not related to any *variant* represents a commonality for the SPL (multiplicity $0 \cdots n$). Moreover, a *variant* must be related to at least one *artefact* (multiplicity $1 \cdots n$). For example, it is possible to trace all the affected security requirements artefacts for a given variant or variation point or vice-versa. This is shown in Fig. 5, in which all the security requirements artefacts affected by the variation point “user authenticity” can be traced. A *security artefact* must also be categorized. The category attribute (compulsory for every *artefact*) assists in avoiding semantic problems and in the reuse of security artefacts, since these are standardized and in SREPPLine they can only be modified by the product line manager. This is a key element for the Security Requirement Decision Sub-Meta Model because it guides us through the categories, thus permitting the systematic identification of the security requirements artefacts.

As is shown in Fig. 2, the starting points of the Security Requirement Decision Sub-Meta Model are the goals/softgoals and feature models and their correlations, which permit the functional and

Table 7
Task A1.3 definition.

TaskUse: A1.3 **Security Objectives Scoping**

ProcessPerformer {kind: primary}

RoleUse: **Business domain experts** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security assets**
WorkProductUse: **Graph of assets dependences**
WorkProductUse: **Security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable security assets valuated** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}

Steps
Step: A1.3.1 **Select security dimensions according to Magerit**
Step: A1.3.2 **Identify the dimension in which each asset is valuable**
Step: A1.3.3 **Value the security assets for each security dimension according to the cost to the organization of the destruction of each asset**
Step: A1.3.4 **Propagate values through the dependencies graph**
Step: A1.3.5 **Determine security objectives rationale**
Step: A1.3.6 **Inspect security objectives valuation**

Guidance
Guidance {kind: **Checklist**}**: Type of security dimensions according to Magerit**
Guidance {kind: **Checklist**}**: Scale to value assets according to Magerit valuation criteria**
Guidance {kind: **Practice**}**: Attack trees**
Guidance {kind: **Practice**}**: Graph of dependencies of assets**
Guidance {kind: **Practice**}**: Delphi evaluation**
Guidance {kind: **Practice**}**: Meetings**

Table 8
Task A1.4 definition.

TaskUse: A1 **.4 Security Threats Scoping**

ProcessPerformer {kind: primary}

RoleUse: **Business domain experts** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security assets valuated**
WorkProductUse: **Security reference model**
WorkProductUse: **List of public vulnerabilities**
WorkProductUse: **Security / failure / incident reports or logs of existing products**
WorkProductUse: **Security standards (ISO/IEC 27001 control objectives and ISO/IEC 15408 families)**
WorkProductUse: **Organization security policy and architecture**
WorkProductUse: **Product line scenario**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable threats** {state: **initial draft**}
WorkProductUse: **Vulnerabilities report** {state: **initial draft**}
WorkProductUse: **Existing safeguards** {state: **initial draft**}
WorkProductUse: **Safeguards evaluation report** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}

Steps
Step: A1.4.1 **Identify and select security vulnerabilities**
Step: A1.4.2 **Identify type of attackers**
Step: A1.4.3 **Identify type of attacks**
Step: A1.4.4 **Determine the relevant ISO/IEC 27001 control objectives and ISO/IEC 15408 families**
Step: A1.4.5 **Identify common threats for each asset**
Step: A1.4.6 **Identify variable threats for each asset**
Step: A1.4.7 **Categorize threats**
Step: A1.4.8 **Relate each security variable artefacts (misuse cases, attack trees, security standards) to the corresponding variants or variation points in the security reference model**
Step: A1.4.9 **Model and specify threats**
Step: A1.4.10 **Identify existing safeguards**
Step: A1.4.11 **Verify threats and assets against security features and the security reference model**

Guidance
Guidance {kind: **Checklist**}**: Catalogue of type of security threats according to Magerit**
Guidance {kind: **Checklist**}**: Scale to value threats according to Magerit valuation criteria**
Guidance {kind: **Template**}**: Attack trees**
Guidance {kind: **Template**}**: Misuse cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Practice**}**: Delphi evaluation**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**

non-functional requirements to be considered. This is owing to the fact that goal models express high-level intentions regarding the system, which are refined into more concrete requirements, and feature models express the high-level requirements of a system architecture, signifying that both goal models and feature models can be used to express the intentions of a system [53]. The interest in using softgoals and features lies in the fact that, if the traceability links are carefully established, they allow us to decide what security features are needed to maintain the security aligned with the goals of the SPL or product and what the optimal set of security features of a determined priority is in the context of the different scenarios of the SPL that provides the rationale for the selection. This therefore supposes a rise in the abstraction level of the variants selection process, and the selection is made in the requirements level rather than in the design level.

The selected category/ies of each security artefact 'Xi' might allow this Sub-Meta Model to propose (a) category(ies) of the *security artefacts* related to the Xi security artefacts categories, with 'Xi' starting from the selected categories of the security features. This Sub-Meta Model could therefore propose (a) category(ies) of assets related to these categories of security features. The same occurs with threats. That is to say, in the selected category/ies of the asset, along with its/their related security objectives category/ies, this Sub-Meta Model could propose threats or categories of threats related to the assets and security objectives categories in order to assist in the identification and valuation of common and optional threats. This is also true of the security requirements. The selected category/ies of threats could allow this Sub-Meta Model could propose a category or categories of security requirements related to the category/ies which could mitigate the impact or reduce the likelihood of these threats. This mechanism facilitates both the elicitation of the common and optional security requirements of the SPL and the security requirements instantiation in the products.

Finally, in Fig. 2, we have represented the security standards variability by integrating the Common Criteria (CC) elements and the ISO/IEC 27001 controls. These security standard elements must be related to the categories of the security artefact defined in the organization: *security feature*, *threat* and *security requirement*. The aim of this is to assist in the SPL or SPL products' certification with regard to these standards and facilitating their reasoning. For example, the CC class FIA (identification, authentication and binding) can be related to the "authentication" of *security feature* elements category, as occurs in the example represented in Fig. 5.

This Sub-Meta Model facilitates the reasoning of security requirements related artefacts and the security standards' conformance. It supports the capturing, specifying and reasoning of security requirements for both SPLs and SPL members thanks to its trace-links, as will be shown in practice in Section 5 (Fig. 5).

Table 9
Task A1.5 definition.

TaskUse: A1.5 **Security Risk Assessment**

ProcessPerformer {kind: primary}

RoleUse: **Business domain experts** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security assets valuated**
WorkProductUse: **Common and variable threats**
WorkProductUse: **Security reference model**
WorkProductUse: **Organization security policy and architecture**
WorkProductUse: **Existing safeguards**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Potential risk assessment report** {state: **initial draft**}
WorkProductUse: **Residual risk assessment report** {state: **initial draft**}
WorkProductUse: **Deficiencies or weaknesses report of safeguards** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}

Steps
Step: A1.5.1 **Determine the degree of degradation of each threat for each asset in each security objective**
Step: A1.5.2 **Determine the probable frequency of occurrence (likelihood) of each threat for each asset in each security objective**
Step: A1.5.3 **Calculate the impact of each threat for each asset in each security objective**
Step: A1.5.4 **Calculate the potential risk of each threat for each asset in each security objective**
Step: A1.5.5 **Relate each risk value to the corresponding variants or variation points in the security reference model**
Step: A1.5.6 **Propagate values through the dependencies graph**
Step: A1.5.7 **Evaluate effectiveness of existing safeguards**
Step: A1.5.8 **Calculate the residual risk of each threat for each asset in each security objective taking into account the value of the assets and the valuation of threats and the effectiveness of the safeguards currently deployed**
Step: A1.5.9 **Inspect risk assessment and validate risk acceptance**

Guidance
Guidance {kind: **Checklist**}**: Scale to value risks, degradation and frequency of occurrence according to Magerit valuation criteria**
Guidance {kind: **Guideline**}**: Algorithmic analysis according to Magerit techniques**
Guidance {kind: **Practice**}**: Delphi evaluation**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Agreements, contracts and applicable legislation**

Table 10
Task A1.6 definition.

TaskUse: A1.6 **Security Requirements Scoping**

ProcessPerformer {kind: primary}

RoleUse: **Business domain experts** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Product line manager** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security threats modelled**
WorkProductUse: **Security reference model**
WorkProductUse: **Potential and residual risks**
WorkProductUse: **Security standards (ISO/IEC 27001 controls and ISO/IEC 15408 components)**
WorkProductUse: **Product line scenario and context**
WorkProductUse: **Catalogue of product line functional and non-functional requirements**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable security functional and assurance requirements** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}

Steps
Step: A1.6.1 **Identify and select similar security requirements packages**
Step: A1.6.2 **Determine the relevant ISO/IEC 27001 controls and ISO/ IEC 15408 components**
Step: A1.6.3 **Identify common security requirements that mitigate the threats at the level with regard to the risk assessment**
Step: A1.6.4 **Identify variable security requirements that mitigate the threats at the level with regard to the risk assessment**
Step: A1.6.5 **Determine internal variability of elicited security requirements**
Step: A1.6.6 **Categorize security requirements**
Step: A1.6.7 **Determine security requirements dependences with other functional and non-functional requirements**
Step: A1.6.8 **Model security requirements**
Step: A1.6.9 **Relate each security variable artefacts (security use cases, security standards, Protection Profile) to the corresponding variants or variation points in the security reference model**
Step: A1.6.10 **Validate traceability links between security requirements, threats, security objectives and assets**
Step: A1.6.11 **Verify security requirements against security features satisfaction**

Guidance
Guidance {kind: **Template**}**: Security use cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Template**}**: UMLSec**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**

3.4. SREPPLine process

This section describes our process, which is based on the Software Process Engineering Meta-model Specification (SPEM) standard [51] from the Object Management Group (OMG). SPEM is a process meta-model which is used to describe a concrete software development process or a family of related software development process. The SPEM specification is structured as a UML profile, and provides a complete MOF-based meta-model [51]. This meta-process modelling is a type of metamodelling used in software engineering to support the effort of creating flexible process models. The purpose of using process models, and in this case SPEM, is to document and communicate the SREPPLine process, to enhance its reuse and to facilitate its integration into other processes and frameworks. Thus, by using SPEM in the SREPPLine specification we promote the increment of process engineers' productivity and the quality of the global models they produce as a result of the integration of SREPPLine into the process map of their organization or company.

In accordance with SPEM, SREPPLine is described by using the structure shown in Fig. 3. Each activity specifies: *WorkProduct* as both input and output respectively; the roles that perform or participate in this *RoleUse* activity; the collection of *Steps* defined for a *Task Use* that represents all the work that should be carried out to achieve the overall development goal of the *Activity*; and the *Guidance* that specifies the practices, techniques or standards to consider when performing the *Task Use*.

SREPPLine is composed of two activities: the Product Line Security Domain Requirements Engineering (PLSecDomReq) activity (A1) and the Product Line Security Application Requirements Engineering (PLSecAppReq) activity (A2). The structure of these activities is depicted in Fig. 4 through the use of a SPEM 2.0 diagram. As the figure shows, it is an iterative and incremental process, which is an add-in of tasks grouped into two activities that can be incorporated into and tailored to an organization's SPL development process model. No particular order for the steps is specified in the figure because the order in which the steps are performed depends on the particular process that is established in an organization. The activities and their tasks can thus be combined with existing development methods. In addition, not all the tasks and

Table 11
Task A1.7 definition.

TaskUse: A1.7 **Security Requirements Negotiation and Prioritization**
ProcessPerformer {kind: primary}
RoleUse: **Business domain experts** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Product line manager** {kind: in}
RoleUse: **Inspection team** {kind: in}
WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security requirements modelled**
WorkProductUse: **Security reference model**
WorkProductUse: **Potential and residual risks**
WorkProductUse: **Product line roadmap**
WorkProductUse: **Product line goals**
WorkProductUse: **Catalogue of product line functional and non-functional requirements**
WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and variable security functional and assurance requirements prioritized** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}
WorkProductUse: **Security requirements roadmap** {state: **initial draft**}
Steps
Step: A1.7.1 **Prioritize security requirements regarding risks**
Step: A1.7.2 **Select relevant security requirements according to the risk level acceptance for this phase-iteration**
Step: A1.7.3 **Identify conflicts between the relevant security requirements**
Step: A1.7.4 **Identify the relevant security requirements conflicts with other functional or non-functional requirements**
Step: A1.7.5 **Value the cost of implementing the relevant security requirements**
Step: A1.7.6 **Reach trade-offs**
Step: A1.7.7 **Determine final security requirements commonalities**
Step: A1.7.8 **Determine the final priority and roadmap place for each security requirement**
Step: A1.7.9 **Verify inexistence of security requirements prioritization conflicts for this phase-iteration**
Guidance
Guidance {kind: **Template**}**: Security use cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Template**}**: UMLSec**
Guidance {kind: **Guideline**}**: Delphi evaluation**
Guidance {kind: **Guideline**}**: EasyWinWin negotiation]**
Guidance {kind: **Guideline**}**: Grouping prioritization**
Guidance {kind: **Practice**}**: Cost-benefit vs risk analysis**
Guidance {kind: **Practice**}**: Hierarchical requirements representation**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Application requirements matrix**

Table 12
Task A1.8 definition.

TaskUse: A1**.8 Security Requirements Specification**
ProcessPerformer {kind: primary}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}
WorkDefinitionParameter {kind: in}
WorkProductUse: **Common and variable security requirements modelled**
WorkProductUse: **Security reference model**
WorkDefinitionParameter {kind: out}
WorkProductUse: **Security requirements specification** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}
WorkProductUse: **Security requirements rationale** {state: **initial draft**}
WorkProductUse: **Product line Protection Profile** {state: **initial draft**}
WorkProductUse: **Security metrics description** {state: initial draft}
WorkProductUse: **Security requirements acceptance test description** {state: **initial draft**}
WorkProductUse: **Safeguards and countermeasures description** {state: **initial draft**}
Steps
Step: A1.8.1 **Specify security requirements and their variability**
Step: A1.8.2 **Define and specify the product line Protection Profile**
Step: **A1.8.3 Relate each security requirement related artefact to the Protection Profile**
Step: A1.8.4 **Determine security requirements rationale**
Step: A1.8.5 **Describe safeguards and countermeasures**
Step: A1.8.6 **Describe security requirements test acceptance**
Step: A1.8.7 **Describe security requirements metrics**
Step: A1.8.8 **Validate formality of specifications and traceability links**
Step: A1.8.9 **Validate Protection Profile specification against ISO/IEC 15446**
Guidance
Guidance {kind: **Template**}**: Security use cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Template**}**: UMLSec**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**

steps are required, and developers could create their own lightweight process by selecting a subset of the steps in each task.

The following sections explain the most important characteristics of SREPPLine activities. SREPPLine activities are an add-in of tasks but not all the steps in each task are necessary, thus enabling developers to create their own lightweight process by selecting a subset of the steps in each task in order to adapt the process to the size of the project and the organization. In this section, we provide a detailed description of the two activities that we have considered in our process using SPEM 2.0. We define the tasks, roles, steps, work products and guidance of each activity, which will be characterised according to the discipline to which they pertain.

3.4.1. SREPPLine artefacts: roles, work products and guidance

In this section, we provide an overview of the main artefacts (roles, work products and guidance) that SREPPLine uses in its activities.

The guidance (techniques, practices and standards) and work products that SREPPLine uses are common techniques or practices and artefacts of software engineering, SPL engineering or security engineering. Those which are most important and specific are briefly described in Tables 1 and 2.

The roles defined in SREPPLine, and briefly explained in Table 3, supplement the roles already present in software engineering, the difference being that these roles particularly focus on security in SPL and also require special training in some cases. Although one physical person can fulfil multiple roles, such as those concerning security, an acknowledged practice is the separation of duties. This means that the security roles should still be separated from the software engineering roles (the minimum number of physical people will therefore be two), although this may of course be difficult in small organizations.

3.4.2. Activity PLSecDomReq

This sub-section provides details of the formal specification of the SREPPLine PLSecDomReq activity which uses SPEM [51] (Tables 4–14). Its most important characteristics will also be explained.

The main goals of this activity are: the development of common and variable security requirements of the SPL which conform to IEEE 830:1998; their precise documentation in an SPL Protection Profile (PP) adapted document by following the standard ISO/IEC 15446 [24]; and the development of their common and variable related security requirements artefacts.

3.4.2.1. PLSecDomReq tasks. Task A1.1: Security Management Scoping and Security Variability Analysis. In this task the product line manager analyses the requests for additional/altered security features

Table 13
Task A1.9 definition.

TaskUse: A1.9 Security Requirements Artefacts Inspection

ProcessPerformer {kind: primary}

RoleUse: **Inspection team** {kind: in}
RoleUse: **Business domain experts** {kind: in}
RoleUse: **Product line manager** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Residual risk assessment report** {state: **initial draft**}
WorkProductUse: **Deficiencies or weaknesses report** {state: **initial draft**}
WorkProductUse: **Security standards (ISO/IEC 27001 and ISO/IEC 15408) conformance report** {state: **initial draft**}
WorkProductUse: **Security process maturity level report (according to +SAFE)** {state: **initial draft**}
WorkProductUse: **Security requirements specification conformance to IEEE 830 report** {state: **initial draft**}

Steps
Step: A1.9.1 **Verify security requirements satisfy security features at the appropriate assurance level**
Step: A1.9.2 **Calculate the residual risks taking into account the effectiveness of the selected security requirements and the safeguards currently deployed**
Step: A1.9.3 **Validate residual risk acceptance**
Step: A1.9.4 **Verify traceability links consistency**
Step: A1.9.5 **Verify security variability links consistency**
Step: A1.9.6 **Verify security standards (ISO/IEC 27001 and ISO/IEC 15408) conformance**
Step: A1.9.7 **Verify requirements specification against IEEE 830**
Step: A1.9.8 **Verify Protection Profile compliance to Common Criteria (ISO/IEC 15408)**

Guidance
Guidance {kind: **Checklist**}: **ISO/IEC 27001**
Guidance {kind: **Checklist**}: **ISO/IEC 15408**
Guidance {kind: **Checklist**}: **IEEE 830:1998 checklist**
Guidance {kind: **Checklist**}: **Enterprise Assurance Level (EAL) of the Common Criteria (ISO/IEC 15408)**
Guidance {kind: **Practice**}: **Walkthroughs**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Interviews**

Table 14
Second and Next Iterations of PLSecDomReq

Activity (kind: Iteration): Second **Product Line Security Domain Requirements Engineering (PLSecDomReq)**
TaskUse...
Similar to First **Product Line Security Domain Requirements Engineering (PLSecDomReq)** iteration:

- reuse and accumulate existing WorkProductUse assets as input to activities
- change "**initial draft**" output WorkProductUse states with "**revised draft**"

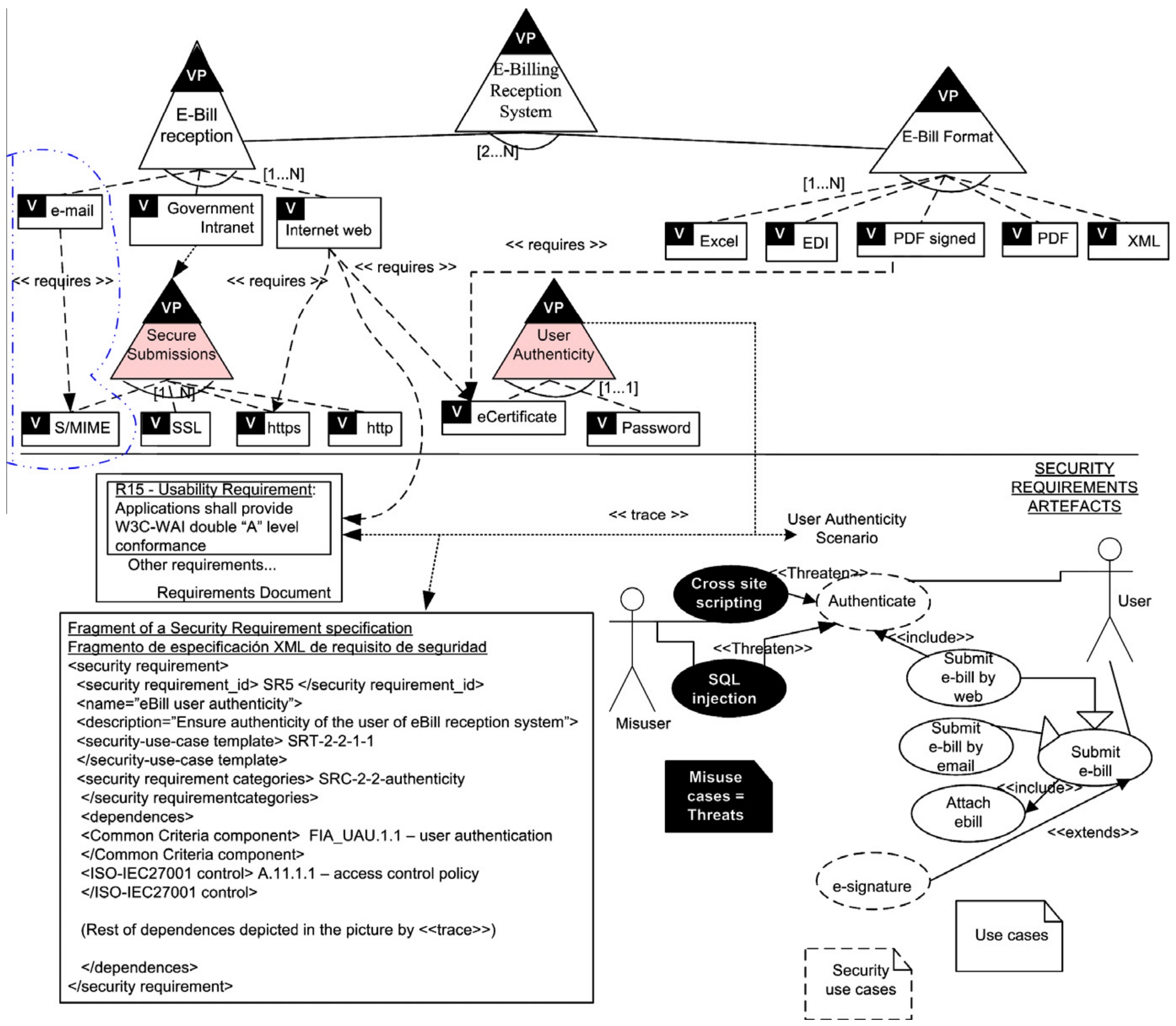
of the SPL. The security goals are defined based on stakeholders' needs, the organization's business goals and security policy, and regulations (laws or internal regulations). The common and variable security features are also identified from the feature model received as an input of this task, and/or new security features are identified simultaneously. They are then all categorized, based on the relevant categories of the Security Reference Model, or new categories are defined if the security features do not match with the existing categories. Examples of these categories are: privacy, access control, etc. The existing similar Protection Profiles in the repository are then identified along with the ISO/IEC 15408 classes and ISO/IEC 27001 clauses. The Security Reference Model of the SPL is next developed and produced as an output of the task in order to specify the variabilities and commonalities and the traceability links of the security features and their related security artefacts.

Task A1.2: Security Assets Scoping. This task identifies the common and variable security assets for each security feature and for the environment, and the dependences between them. The aim of this task is to identify particular components to be developed for reuse, common and variable assets and the dependences between them, and a tree or graph of security assets dependencies is therefore defined. The security assumptions are also set and the stakeholders must reach an agreement regarding the common and optional assets.

Task A1.3: Security Objectives Scoping. The security objectives are the objectives which must be achieved in order to protect the organization's business goals. The security objective categories managed by the meta-model are first selected according to the risk management methodology that SREPPLine is based on (that is, Magerit [37]). The security objective categories managed by the meta-model can therefore only be the following (): *integrity (I), confidentiality (C), availability (D), authenticity of service users (A_S), authenticity of data origin (A_D), accountability of service use (T_S) and accountability of data access (T_D)*. In accordance with Magerit, all the possible security objectives are grouped in seven dimensions, which are the similar term for SREPPLine security objectives categories. This signifies that each security objective must be categorized with one of these categories, since SREPPLine is based on Magerit. The security objectives for each security asset are then stated, and they are also correctly categorized with one of the former security objective categories. The security assets valuation against their related security objectives is next performed, together with a commonality and variability analysis. Each *asset* could have different related security objectives, which are associated with a security objective category (or security dimensions in Magerit [37] terminology). The corresponding *value* is assigned to this category which is agreed by the stakeholders (following a standardized scale from 0 to 10 in accordance with the Magerit risk methodology [37]). This step is performed by carrying out interviews with the different stakeholders, and the Delphi evaluation method [35] and the value scale proposed in Magerit are also used.

The valuation of each asset is given in each security objective category and is propagated through the dependency tree of assets, and it is therefore only necessary to explicitly value the higher assets in the dependency tree. For example, let us assume that the asset "electronic bill", which is related to the security objective "confidentiality of communications" is categorized with 'confidentiality', and that this category is assigned a value of 7, and this asset depends on the asset "web-server", which implies that the "web-server" value for the security objectives categorized with 'confidentiality' will be at least 7 (the propagated value). That is, the accumulated value over an asset is defined as the highest value among it and any of those above. In practice, the result of this step is partially shown in Fig. 6, in which the first number of each cell is the value of the assets. If the numbers are between brackets they are propagated values.

Task A1.4: Security Threats Scoping. The assets are exposed to threats which may prevent the security objective from being achieved. Not all threats affect all assets or all their security objectives, so those which are common and optional must be identified in this task. There is also a certain relationship between the category of the asset and what might happen to it. The Security Reference Model could, therefore, use the selected asset category/ies, along with its/their related security objective category/ies to propose threats or threat categories related to these asset and security objective categories in order to assist in the identification and evaluation of common and optional threats. For example, the "Entry of incorrect information" and "Information alteration" *threat* categories can be related to the "data" or "information" *asset* categories and the "integrity" *security objective* category. The relevant ISO/IEC 27001 control objectives and ISO/IEC 15408 families related



R15 - Usability Requirement: Applications shall provide W3C-WAI double "A" level conformance

Other requirements...

Fragment of a Security Requirement specification

Fragmento de especificación XML de requisito de seguridad

```
<security requirement>
  <security requirement_id> SR5 </security requirement_id>
  <name="eBill user authenticity">
  <description="Ensure authenticity of the user of eBill reception system">
  <security-use-case template> SRT-2-2-1-1
  </security-use-case template>
  <security requirement categories> SRC-2-2-authenticity
   </security requirementcategories>
   <dependences>
   <Common Criteria component>  FIA_UAU.1.1 – user authentication
   </Common Criteria component>
   <ISO-IEC27001 control> A.11.1.1 – access control policy
   </ISO-IEC27001 control>

   (Rest of dependences depicted in the picture by <<trace>>)

   </dependences>
</security requirement>
```

Fig. 5. Example: Part of the eBill-PL Security Reference Model and security artefacts.

Table 15
Activity 1 (PLSecAppReq) definition.

Activity {kind = Phase}: **Application Requirements Engineering** Process: **SREPPLine**
Activity {kind = Iteration}: First **Product Line Security Application Requirements Engineering (PLSecAppReq)**

to the SPL threats are also set. The potential vulnerabilities in public domain sources are revised and the identification of the attack tree [52] associated with the business pattern or SPL domain and the threat modelling and specification are carried out (misuse cases [59] or aspect-oriented XML [32]). Finally, the validation of security goals against threats and assets and their Security Reference Model is carried out.

Task A1.5: Security Risks Assessment. This task consists of the following major steps in order to achieve 100% risk acceptance: assessing whether the threats are relevant according to the security level specified by the security objectives; estimating the security risks based on the relevant threats, their likelihood and their potential negative impacts, depending on the variation points. To do this, we propose the use of Magerit [37], which conforms to ISO/IEC 13335 (GMITS) [23], and which is a methodology officially recognised by NATO at the 9th NATO cyberdefense workshop in 2008 and by OECD [49]. The impact of each threat is calculated by considering the values of the assets of each security objective along with the degradation caused by the threat, which must be estimated by the security risk expert within a range from 0% to 100% (Impact = round(accumulated value $\times$ degradation)). The impact and the likelihood of occurrence of the threat, which must also be estimated by the security risk expert, are also considered in order to calculate the risk according to a formula defined in Magerit ($\Re(\mathrm{Vi}, \mathrm{Fj}) = \mathrm{Vi} + \mathrm{j} - \mathrm{n})$). The risk is then classified in a range of 0–5 according to the Magerit [37] scale. All these data are stored in the Security Requirement Decision Sub-Meta Model and are presented in a table. An example of this table is shown in Fig. 6, in which high values indicate high impacts and risks. SREPPLineTool is also capable of automatically calculating impacts and risks by introducing the degradation and likelihood of occurrence, as can be seen in the example shown in Fig. 7).

Table 16
Task A2.1 definition.

TaskUse: A2.1 **Application Security Variability Management**

ProcessPerformer {kind: primary}

RoleUse: **Product line manager** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Stakeholders of the application needs**
WorkProductUse: **Application specific environment, policies and regulations**
WorkProductUse: **Security standards (ISO/IEC 27001 clauses and ISO/IEC 15408 classes)**
WorkProductUse: **Product line security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Instantiated application security features** {state: **initial draft**}
WorkProductUse: **Application's stakeholder security features that do not correspond to domain security features** {state: **initial draft**}
WorkProductUse: **Application security variability sub- model** {state: **initial draft**}
WorkProductUse: **Application security requirement decision sub-model** {state: **initial draft**}

Steps
Step: A2.1.1 **Define security goals of the application based on stakeholders needs, business goals, security policy and regulations**
Step: A2.1.2 **Communicate the relevant variation points, variants and their dependencies of the domain security features to the stakeholders of the application**
Step: A2.1.3 **Analyze the common security artefacts (security features, assets, security objectives, threats and security requirements) of the product line**
Step: A2.1.4 **Determine the relevant ISO/IEC 27001 clauses and ISO/ IEC 15408 classes for the application**
Step: A2.1.5 **Identify similar product Security Targets among the applications of the product line**
Step: A2.1.6 **Analyze the security features variability (Analyze the variants of each variation point)**
Step: A2.1.7 **Select the domain security features variants that satisfy the security features of the stakeholders of the application**
Step: A2.1.8 **Trace each application security feature variant to the corresponding domain variants or variation points in the security reference model of the product line**
Step: A2.1.9 **Model the security reference model of the application**
Step: A2.1.10 **Collect the application's stakeholder security features that do not correspond to domain security features**
Step: A2.1.11 **Inspect the selected security features for the application and the application security reference model**

Guidance
Guidance {kind: **Practice**}**: Questionnaire**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Hierarchical security features**
Guidance {kind: **Checklist**}**: Organization policy, laws and standards**

Table 17
Task A2.2 definition.

TaskUse: A2.2 **Application Security Artefacts Instantiation**

ProcessPerformer {kind: primary}

RoleUse: **Product line manager** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Selected – instantiated security features**
WorkProductUse: **Application security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Common and instantiated assets** {state: **initial draft**}
WorkProductUse: **Common and instantiated security objectives** {state: **initial draft**}
WorkProductUse: **Common and instantiated threats** {state: **initial draft**}
WorkProductUse: **Common and instantiated security requirements** {state: **initial draft**}
WorkProductUse: **Common and instantiated security metrics description** {state: **initial draft**}
WorkProductUse: **Common and instantiated security requirements acceptance test description** {state: **initial draft**}
WorkProductUse: **Common and instantiated safeguards and countermeasures description** {state: **initial draft**}
WorkProductUse: **Application security reference model** {state: **initial draft**}

Steps
Step: A2.2.1 **Analyze the assets variability**
Step: A2.2.2 **Select the domain assets variants that satisfy the security features of the stakeholders of the application**
Step: A2.2.3 **Analyze the security objectives variability**
Step: A2.2.4 **Select the appropriate variable security objectives from the product line for the application**
Step: A2.2.5 **Analyze the threats variability**
Step: A2.2.6 **Select the appropriate variable threats from the product line for the application**
Step: A2.2.7 **Analyze the security requirements variability**
Step: A2.2.8 **Select the appropriate variable security requirements from the product line for the application**
Step: A2.2.9 **Trace each application security feature variant to the corresponding domain variants or variation points in the security reference model of the product line**
Step: A2.2.10 **Model the security reference model of the application**
Step: A2.2.11 **Inspect the common and selected security artefacts and the application security reference model**

Guidance
Guidance {kind: **Template**}**: Attack trees**
Guidance {kind: **Template**}**: Misuse cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Template**}**: Security use cases**
Guidance {kind: **Template**}**: Aspect XML**
Guidance {kind: **Template**}**: UMLSec**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**

Task A1.6: Security Requirements Scoping. The Security Reference Model is able to propose a category or categories of security requirements related to all the selected threat category/ies which could mitigate the impact or reduce the likelihood of these threats. This mechanism facilitates both the elicitation of the common and optional security requirements of the SPL and the security requirements instantiation in the products. The misuse cases and their related threats are therefore first analyzed in this task, and the appropriate CC (ISO/IEC 15408) security functional requirements and ISO/IEC 27001 controls for the threats of the product line are then selected. The identification of the common security requirements according to the elicited requirements and obtained through the previously performed risk analysis are next carried out. This leads to the definition of the variable security requirements and variability dependencies between them, while a CC security requirement is simultaneously defined which permits CC operations (iteration, assignment, selection or refinement). Finally the security requirements are modelled with security use cases [18] or UMLSec [28] and are traced with their associated security test and security measure/metric. Examples of security use cases are shown in Fig. 5, and are depicted with a dotted line.

Task A1.7: Security Requirements Negotiation and Prioritization. This task comprises the following major steps: identification of the interdependences with other functional and non-functional requirements and trade-offs in the security requirements decision model; and balancing the risk with the economical impact of implementing countermeasures. Various techniques, such as Easy-WinWin negotiation [21] and grouping prioritization [8] are used.

Table 18
Task A2.3 definition.

TaskUse: A2.3 **Sec-Deltas Analysis and Application Specific Security Artefacts Development**

ProcessPerformer {kind: primary}

RoleUse: **Product line manager** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Selected – instantiated security artefacts**
WorkProductUse: **Application security reference model**
WorkProductUse: **Application's stakeholder security features that do not correspond to domain security features**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Sec-deltas and their impact** {state: **initial draft**}
WorkProductUse: **Application specific security artefacts** {state: **initial draft**}
WorkProductUse: **Application security reference model** {state: **initial draft**}

Steps
Step: A2.3.1 **Analyze sec-deltas**
Step: A2.3.2 **Analyze the impact on the security reference model (necessity of adding new security variants or variation points for each security artefact)**
Step: A2.3.3 **Analyze the impact of the security reference model sec- deltas on the corresponding security artefacts (Changes in variation points, variants, dependencies on associated variants)**
Step: A2.3.4 **Identify and develop application specific security artefacts.**
Inherited steps from:
⇒ TaskUse: A1.2 **Security Assets Scoping**
⇒ TaskUse: A1.3 **Security Objectives Scoping**
⇒ TaskUse: A1.4 **Security Threats Scoping**
⇒ TaskUse: A1.5 **Security Risk Assessment**
⇒ TaskUse: A1.6 **Security Requirements Scoping**
Step: A2.3.5 **Update the security reference model of the application and their traceability and variability links**
Step: A2.3.6 **Inspect the sec-deltas and the application specific security artefacts and the application security reference model**

Guidance
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Questionnaire**
Guidance used in the inherited steps

Task A1.8: Security Requirements Specification. This task consists of security requirements modelling and security requirements specification and the Product Line Protection Profile specification according to ISO/IEC 15446 [24]. In order to do this we use the technique of security use cases and their parametrical templates, which are traced to the security variability model. Fig. 5 shows an example of part of a security requirement specification with the XML aspect requirements specification technique, and with its traces to the security variability model.

Task A1.9: Security Requirements Artefacts Inspection. This task comprises the following major steps: (i) it is first necessary to verify whether the security requirements satisfy the stakeholders' security needs and the SPL security goals, and also whether a risk assessment outlined to obtain the residual risks has been performed; (ii) we verify whether the security requirements conform to ISO/IEC 27001 control objectives, to CC (ISO/IEC 15408) assurance requirements and to the IEEE 830-1998 standard since, according to this standard, a quality requirement must be correct, unambiguous, complete, consistent, ranked by importance and/or stability, verifiable, modifiable, and traceable; (iii) we also propose the use of the CMMI-DEV+SAFE [57] in order to assist in the evaluation of the product line security engineering process in the Domain Testing sub-process; and (iv) we verify the fulfilment of the previously approved EAL and the CC evaluation.

Table 19
Task A2.4 definition.

TaskUse: A2.4 **Application Risk Assessment**

ProcessPerformer {kind: primary}

RoleUse: **Expert users** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Application security assets valuated**
WorkProductUse: **Application threats**
WorkProductUse: **Application security reference model**
WorkProductUse: **Organization security policy and architecture**
WorkProductUse: **Existing safeguards**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Potential risk assessment report** {state: **initial draft**}
WorkProductUse: **Residual risk assessment report** {state: **initial draft**}
WorkProductUse: **Sec-deltas risk assessment report** {state: **initial draft**}
WorkProductUse: **Deficiencies or weaknesses report of safeguards** {state: **initial draft**}
WorkProductUse: **Security reference model** {state: **initial draft**}

Steps
⇒ Inherited steps from: TaskUse: A1.5 Security risk assessment
Step: A2.4.1 **Determine the degree of degradation of each threat for each asset in each security objective**
Step: A2.4.2 **Determine the probable frequency of occurrence (likelihood) of each threat for each asset in each security objective**
Step: A2.4.3 **Calculate the impact of each threat for each asset in each security objective**
Step: A2.4.4 **Calculate the potential risk of each threat for each asset in each security objective**
Step: A2.4.5 **Relate each risk value to the corresponding variants or variation points in the security reference model**
Step: A2.4.6 **Propagate values through the dependencies graph**
Step: A2.4.7 **Evaluate effectiveness of existing safeguards**
Step: A2.4.8 **Calculate the residual risk of each threat for each asset in each security objective taking into account the value of the assets and the valuation of threats and the effectiveness of the safeguards currently deployed**
Step: A2.4.9 **Inspect risk assessment and validate risk acceptance**

Guidance
Guidance {kind: **Checklist**}**: Scale to value risks, degradation and frequency of occurrence according to Magerit valuation criteria**
Guidance {kind: **Guideline**}**: Algorithmic analysis according to Magerit techniques**
Guidance {kind: **Practice**}**: Delphi evaluation**
Guidance {kind: **Practice**}**: Meetings**
Guidance {kind: **Practice**}**: Interviews**
Guidance {kind: **Practice**}**: Agreements, contracts and applicable legislation**

Subsequent iterations. The subsequent iterations will be specified in a similar manner to that of the aforementioned activities specification, as is shown in Table 14.

3.4.3. Activity PLSecAppReq

This section provides details of the formal specification of the SREPPLine PLSecAppReq activity using SPEM [51] (Tables 12–20). Its most important characteristics will also be explained.

The main goals of this sub-process are: elicitation and documentation of the security requirements and their related security artefacts in the SPL application; ensuring that they conform to IEEE 830:1998 and gathering them together in a Security Targets (ST) adapted document by following the ISO/IEC 15446 [24] standard, reusing the security domain artefacts and requirements as far as possible.

3.4.3.1. PLSecAppReq tasks. The PLSecAppReq *task A2.1* is that of "*Application Security Variability Management*". In this task stakeholders are informed of the commonalities and variabilities of the security features of the SPL, since the goal of this task is both

Table 20
Task A2.5 definition.

TaskUse: A2.5 **Application Security Requirements Negotiation and Prioritization**

ProcessPerformer {kind: primary}

RoleUse: **Expert users** {kind: in}
RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Requirements engineer** {kind: in}
RoleUse: **Security expert** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Product line manager** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Application security requirements modelled**
WorkProductUse: **Application security reference model**
WorkProductUse: **Potential and residual risks**
WorkProductUse: **Application roadmap**
WorkProductUse: **Application goals**
WorkProductUse: **Application functional and non-functional requirements**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Application security functional and assurance requirements prioritized** {state: **initial draft**}
WorkProductUse: **Application security reference model** {state: **initial draft**}
WorkProductUse: **Application security requirements roadmap** {state: **initial draft**}
WorkProductUse: **Request for additional / altered security domain artefacts in the product line** {state: **initial draft**}

Steps
Step: A2.5.1 **Analyze realisation effort of each variability model delta and their respective security requirements related artefacts deltas**
Step: A2.5.2 **Prioritize security requirements regarding risks**
Step: A2.5.3 **Select relevant security requirements according to the risk level acceptance for this phase-iteration**
Step: A2.5.4 **Identify conflicts between the relevant security requirements**
Step: A2.5.5 **Identify conflicts about the deltas realisation effort in the application team development or in product line team development**
Step: A2.5.6 **Identify the relevant security requirements conflicts with other functional or non-functional requirements**
Step: A2.5.7 **Value the cost and effort of implementing the relevant security requirements**
Step: A2.5.8 **Reach trade-offs regarding the relevant security requirements and realisation effort**
Step: A2.5.9 **Determine final security requirements artefacts deltas realisation effort decision (realised, partial realisation, stakeholders adapts requirement, removed stakeholder requirement) and the team development (application team or product line team)**
Step: A2.5.10 **Determine the final priority and roadmap place for each security requirement**
Step: A2.5.11 **Verify inexistence of security requirements prioritization conflicts for this phase-iteration**

Guidance
Guidance {kind: **Template**}: **Security use cases**
Guidance {kind: **Template**}: **Aspect XML**
Guidance {kind: **Template**}: **UMLSec**
Guidance {kind: **Guideline**}: **Delphi evaluation**
Guidance {kind: **Guideline**}: **EasyWinWin negotiation**
Guidance {kind: **Guideline**}: **Grouping prioritization**
Guidance {kind: **Practice**}: **Cost/effort-benefit** vs. **risk analysis**
Guidance {kind: **Practice**}: **Hierarchical requirements representation**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Interviews**
Guidance {kind: **Practice**}: **Matrix of sec-deltas categories (new variant, new variation point, adapt dependencies) vs adaptation effort categories (no, moderate, high, too high)**

Table 21
Task A2.6 definition.

TaskUse: A2.6 **Application Security Requirements Specification**

ProcessPerformer {kind: primary}

RoleUse: **Security requirements engineer** {kind: in}
RoleUse: **Security architect** {kind: in}
RoleUse: **Inspection team** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Application security requirements modelled**
WorkProductUse: **Application security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Security requirements specification** {state: **initial draft**}
WorkProductUse: **Application security reference model** {state: **initial draft**}
WorkProductUse: **Security requirements rationale** {state: **initial draft**}
WorkProductUse: **Application Security Target** {state: **initial draft**}
WorkProductUse: **Security metrics description** {state: **initial draft**}
WorkProductUse: **Security requirements acceptance test description** {state: **initial draft**}
WorkProductUse: **Safeguards and countermeasures description** {state: **initial draft**}

Steps
Step: A2.6.1 **Specify security requirements**
Step: A2.6.2 **Specify application security requirements artefacts that correspond to security domain requirements artefacts including their traces**
Step: A2.6.3 **Specify the application variability model with the selected variants**
Step: A2.6.4 **Specify the application variability model deltas including the traces to the original variability model elements of the product line**
Step: A2.6.5 **Specify the traces between the security requirements and the variants selected for the application**
Step: A2.6.6 **Specify the traces of the risks and realisation costs to the sec-deltas to sec-deltas decisions traceability**
Step: A2.6.7 **Define and specify the application Security Target**
Step: A2.6.8 **Relate each security requirement related artefact to the Security Target**
Step: A2.6.9 **Determine security requirements rationale**
Step: A2.6.10 **Describe safeguards and countermeasures**
Step: A2.6.11 **Describe security requirements test acceptance**
Step: A2.6.12 **Describe security requirements metrics**
Step: A2.6.13 **Validate formality of specifications and verify traceability links and variability links**
Step: A2.6.14 **Validate Security Target specification against ISO/IEC 15446**

Guidance
Guidance {kind: **Template**}: **Security use cases**
Guidance {kind: **Template**}: **Aspect XML**
Guidance {kind: **Template**}: **UMLSec**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Interviews**

to make the stakeholders aware of the security goals and features of the SPL and to elicit application security goals and features. The Security Requirements Decision Model and the Security Variability Model enable the security requirements engineer to communicate the relevant security related variation points, security related variants and their dependences to stakeholders. The variability model's traceability links to security domain artefacts also enable the security requirements engineer to describe the particularities of a particular security related variant. Once the stakeholders have informed the security requirements engineer of their security goals and of the features necessary for the application (or product), the result of this task is a set of domain security goals and features of the SPL, which may not completely fulfil the stakeholders' security goals for the application.

In *task A2.2* ("*Application Security Artefacts Instantiation*") application security artefacts from the set of domain security features obtained in the previous task are instantiated. The appropriate security artefacts, i.e., the security variants, for the specific application (product) that will as far as possible satisfy the application security goals are selected from the Security Requirements Decision Model and the Security Variability Model. The result of this task is a set of security requirements and their related artefacts, which may not completely fulfil the stakeholders' application requirements.

In *task A2.3* "*Sec-Deltas Analysis and Application Specific Security Artefacts Development*" the sec-deltas analysis is performed. The sec-deltas occur when stakeholder security requirements cannot

Table 22
Task A2.7 definition.

TaskUse: A2.7 **Application Security Requirements Artefacts Inspection**

ProcessPerformer {kind: primary}

RoleUse: **Inspection team** {kind: in}
RoleUse: **Expert users** {kind: in}
RoleUse: **Product line manager** {kind: in}

WorkDefinitionParameter {kind: in}
WorkProductUse: **Application security reference model**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Residual risk assessment report** {state: **initial draft**}
WorkProductUse: **Deficiencies or weaknesses report** {state: **initial draft**}
WorkProductUse: **Security standards (ISO/IEC 27001 and ISO/IEC 15408)conformance report** {state: **initial draft**}
WorkProductUse: **Security process maturity level report (according to +SAFE)** {state: **initial draft**}
WorkProductUse: **Security requirements specification conformance to IEEE 830 report** {state: **initial draft**}

Steps
Step: A2.7.1 **Verify security requirements satisfy security features at the appropriate assurance level**
Step: A2.7.2 **Calculate the residual risks taking into account the effectiveness of the selected security requirements and the safeguards currently deployed**
Step: A2.7.3 **Validate residual risk acceptance**
Step: A2.7.4 **Verify traceability links consistency**
Step: A2.7.5 **Verify security variability links consistency**
Step: A2.7.6 **Verify security standards (ISO/IEC 27001 and ISO/IEC 15408) conformance**
Step: A2.7.7 **Verify requirements specification against IEEE 830**
Step: A2.7.8 **Verify Security Target compliance to Common Criteria (ISO/IEC 15408)**
Step: A2.7.9 **Analyze the submission of the requests of adding / altered security artefacts in the product line**

Guidance
Guidance {kind: **Checklist**}: **ISO/IEC 27001**
Guidance {kind: **Checklist**}: **ISO/IEC 15408**
Guidance {kind: **Checklist**}: **IEEE 830:1998 checklist**
Guidance {kind: **Checklist**}: **Enterprise Assurance Level (EAL) of the Common Criteria (ISO/IEC 15408)**
Guidance {kind: **Practice**}: **Walkthroughs**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Interviews**

Table 23
Second and Next Iterations of PLSecAppReq.

Activity (kind: Iteration): Second **Product Line Security Application Requirements Engineering (PLSecAppReq)**
TaskUse...
Similar to First **Product Line Security Application Requirements Engineering (PLSecAppReq)** iteration:

- reuse and accumulate existing WorkProductUse assets as input to activities
- change <<**initial draft**>> output WorkProductUse states with <<**revised draft**>>

be completely satisfied by security domain requirements artefacts. Sec-deltas to the security domain variability model resulting from stakeholders' security features/goals are analyzed during the sec-deltas analysis. Next, the impact of the security variability model sec-deltas on the corresponding security artefacts is analyzed. The result of this analysis is the security application variability model, along with the security requirements artefacts deltas which are developed immediately afterwards. Finally, these sec-deltas are communicated to the security risk expert who estimates the risks involved in carrying out the security requirements deltas or otherwise (*task A2.4* "*Application Risk Assessment*").

The "*Application Security Requirements Negotiation and Prioritization*" task (*task A2.5* of PLSecAppReq). After the application risk assessment of the sec-deltas has been performed, its results are communicated to the security architect and to the security requirements engineer who estimate the realisation effort based on the sec-deltas and their associated risks. The stakeholders then use this estimation to decide whether or not the security requirements deltas should be carried out and which security standard the application should fulfil. The application security requirements and the corresponding security requirements artefacts and security application variability model are defined as a result of this task.

In the "*Application Security Requirements Specification*" task (*task A2.6* of PLSecAppReq) the application security artefacts, the sec-deltas and the traces between application security artefacts and the corresponding domain security artefacts are specified and documented. The security application variability model and the traceability links of the application security artefacts to the application-specific variability model are also documented. The estimated risk and realisation costs are even related to the sec-deltas to ensure that decisions regarding sec-deltas are traceable.

Finally, in *task A2.7* ("*Application Security Requirements Inspection*"), the same points listed in the PLSecDomReq task A1.9 (Security Requirements Artefacts Inspection) are verified along with the security requirements artefacts variability consistency between the application and domain artefacts. The product line manager also evaluates the Sec-Deltas in order to decide which security features, security requirements, in addition to their related security artefacts (assets, security objectives, threats, countermeasures, etc.), and their traceability and variability links should be part of the product line domain artefacts.(See Tables 21 and 22).

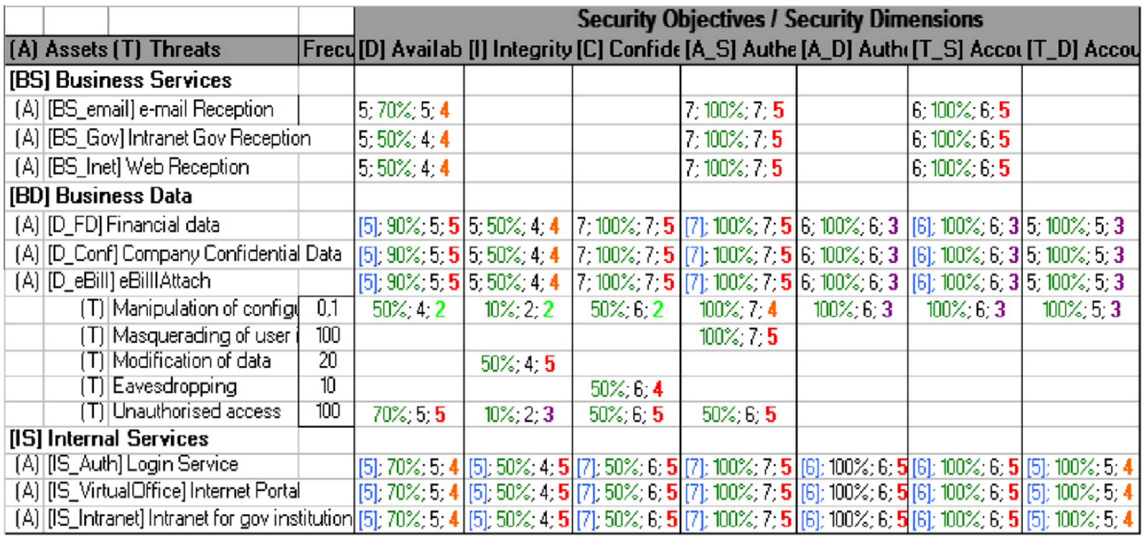
Subsequent iterations. The subsequent iterations will be specified in a similar manner to that of the aforementioned activities specification, as is shown in Table 23.

4. Case study: Applying SREPPLine

In this section we shall illustrate the application of SREPPLine, the Security Reference Meta Model and SREPPLineTool applicability to security requirements artefacts management in SPL engineering with an e-billing reception service product line for Spanish public administrations (eBill-PL). This SPL may have several different configurations for various public institutions within Spanish Public Administration. It has a common set of system functionalities that forms the deliverable core, and a variable set of configurable parameters and non-functional requirements. eBill-PL (e-billing reception service product line for Spanish public administrations) is therefore an SPL whose members vary through system configuration and online business services and yet retain the same core functionalities.

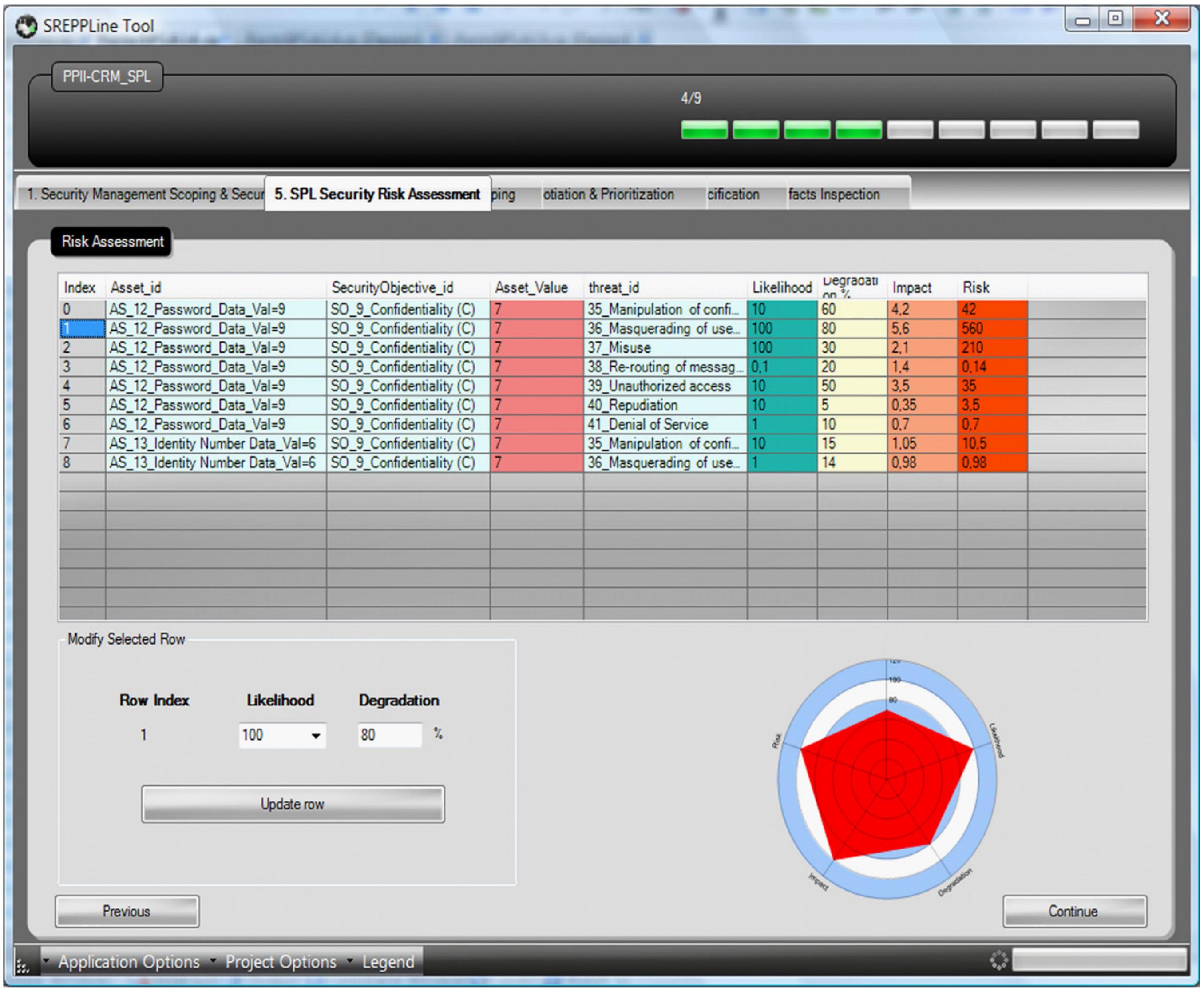
This example concentrates on the results obtained from the application of PLSecAppReq (SREPPLine task) to application engineering in order to develop an e-billing reception service platform within the official website of a Spanish Public Administration from the e-billing reception service product line for Spanish public administrations (named eBill-PL). It is focused on the security features of the e-billing reception/submission service of the e-billing platform. It was necessary to simplify and summarize this example (basically the output and input artefacts generated in each task have been omitted or briefly/partially described) in order to facilitate the illustration of points of the SREPPLine process in this paper.

The eBill-PL provides the variability as represented by the variability model in Fig. 5. It offers different variants (V), which in this example are variation points (VP), for the different public functionalities. These are the business services offered by the e-billing reception service product line to companies and citizens, which could be selected by the application stakeholder. During PLSecApp-



		Security Objectives / Security Dimensions						
(A) Assets (T) Threats	Frecu	[D] Availab	[I] Integrity	[C] Confide	[A_S] Authe	[A_D] Authe	[T_S] Accou	[T_D] Accou
[BS] Business Services								
(A) [BS_email] e-mail Reception		5; 70%; 5; 4			7; 100%; 7; 5		6; 100%; 6; 5	
(A) [BS_Gov] Intranet Gov Reception		5; 50%; 4; 4			7; 100%; 7; 5		6; 100%; 6; 5	
(A) [BS_Inet] Web Reception		5; 50%; 4; 4			7; 100%; 7; 5		6; 100%; 6; 5	
[BD] Business Data								
(A) [D_FD] Financial data		[5]; 90%; 5; 5	5; 50%; 4; 4	7; 100%; 7; 5	[7]; 100%; 7; 5	6; 100%; 6; 3	[6]; 100%; 6; 3	5; 100%; 5; 3
(A) [D_Conf] Company Confidential Data		[5]; 90%; 5; 5	5; 50%; 4; 4	7; 100%; 7; 5	[7]; 100%; 7; 5	6; 100%; 6; 3	[6]; 100%; 6; 3	5; 100%; 5; 3
(A) [D_eBill] eBillAttach		[5]; 90%; 5; 5	5; 50%; 4; 4	7; 100%; 7; 5	[7]; 100%; 7; 5	6; 100%; 6; 3	[6]; 100%; 6; 3	5; 100%; 5; 3
(T) Manipulation of configu	0,1	50%; 4; 2	10%; 2; 2	50%; 6; 2	100%; 7; 4	100%; 6; 3	100%; 6; 3	100%; 5; 3
(T) Masquerading of user i	100				100%; 7; 5			
(T) Modification of data	20		50%; 4; 5					
(T) Eavesdropping	10			50%; 6; 4				
(T) Unauthorised access	100	70%; 5; 5	10%; 2; 3	50%; 6; 5	50%; 6; 5			
[IS] Internal Services								
(A) [IS_Auth] Login Service		[5]; 70%; 5; 4	[5]; 50%; 4; 5	[7]; 50%; 6; 5	[7]; 100%; 7; 5	[6]; 100%; 6; 5	[6]; 100%; 6; 5	[5]; 100%; 5; 4
(A) [IS_VirtualOffice] Internet Portal		[5]; 70%; 5; 4	[5]; 50%; 4; 5	[7]; 50%; 6; 5	[7]; 100%; 7; 5	[6]; 100%; 6; 5	[6]; 100%; 6; 5	[5]; 100%; 5; 4
(A) [IS_Intranet] Intranet for gov institution		[5]; 70%; 5; 4	[5]; 50%; 4; 5	[7]; 50%; 6; 5	[7]; 100%; 7; 5	[6]; 100%; 6; 5	[6]; 100%; 6; 5	[5]; 100%; 5; 4

Fig. 6. Part of the risk assessment of the eBill-PL.



Index	Asset_id	SecurityObjective_id	Asset_Value	threat_id	Likelihood	Degradation %	Impact	Risk
0	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	35_Manipulation of confi...	10	60	4,2	42
1	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	36_Masquerading of use...	100	80	5,6	560
2	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	37_Misuse	100	30	2,1	210
3	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	38_Re-routing of messag...	0,1	20	1,4	0,14
4	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	39_Unauthorized access	10	50	3,5	35
5	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	40_Repudiation	10	5	0,35	3,5
6	AS_12_Password_Data_Val=9	SO_9_Confidentiality (C)	7	41_Denial of Service	1	10	0,7	0,7
7	AS_13_Identity Number Data_Val=6	SO_9_Confidentiality (C)	7	35_Manipulation of confi...	10	15	1,05	10,5
8	AS_13_Identity Number Data_Val=6	SO_9_Confidentiality (C)	7	36_Masquerading of use...	1	14	0,98	0,98

Modify Selected Row

Row Index	Likelihood	Degradation
1	100	80 %

Update row

Fig. 7. Table 8 of SREPPLineTool: Risk Assessment.

Req task A2.1 (“*Application Security Variability Management*”), both the Security Requirements Decision Model and the Security Variability Model had enabled the security requirements engineer to communicate the relevant security related variations points (VP), security related variants and their dependences (security artefacts, security standards and other functional and non-functional requirements) to the stakeholders. The stakeholders informed the security requirements engineer of their security goals and of the features necessary for the application (or product). The result of this task was a set of domain security goals and features of the SPL which did not completely fulfil the stakeholders' security goals for the application.

In this example we selected the following security features: user authenticity and secure submissions. As is shown in Fig. 5, different authenticity methods are selectable from the eBill-PL for the ‘user authenticity’ variation point. It offers the following security variants: ‘password’ and ‘electronic certificate’. Four security variants can be selected for the ‘secure submissions’ variation point: ‘http’, ‘SSL’, ‘https’ and ‘S/MIME’.

In task A2.2 of PLSecAppReq (“*Application Security Artefacts Instantiation*”), application security artefacts from the set of domain security features obtained in the previous task were instantiated. The appropriate security artefacts, i.e., the security variants, for the specific application (product) which would as far as possible satisfy the application goals and softgoals were selected from the Security Requirements Decision Sub-Model and the Security Variability Sub-Model. These had already been instantiated from the respective previously explained meta-models. The result of this task was a set of security requirements and their related artefacts which did not completely fulfil the stakeholders' application requirements. In this example, at the ‘secure submissions’ VP we selected the ‘https’ security variant because the stakeholders had selected the ‘Internet web’ variant and owing to the security links (or traceability links) established in the Security Requirements Decision Sub-Model of the e-billing reception service product line. At the ‘user authenticity’ VP we selected the ‘e-certificate’ security variant because the stakeholders had selected the ‘email’ and ‘Internet web’ feature for the e-billing reception service of the Institution.

In task A2.3 “*Sec-Deltas Analysis and Application Specific Security Artefacts Development*” the sec-deltas analysis was performed. The sec-deltas occur when stakeholder security requirements cannot be completely satisfied by security domain requirements artefacts. Sec-deltas to the security domain variability model resulting from stakeholders' security features were analyzed during the sec-deltas analysis. The particular stakeholders' needs for the e-billing reception service of the Institution made it necessary to add one more variant to the ‘E-Bill Reception’ variation point regarding the reception of e-bills by email (‘email’ variant). This type of request necessitated a different kind of secure submission mechanism which is not among the variants available in the eBill-PL. We therefore identified one sec-delta (depicted as a discontinued line in Fig. 5) since the SPL did not provide any security features to ensure secure emails submission. We then added one more security variant for the ‘Secure submissions’ to the application variability model, as is shown in Fig. 5. This variant was: ‘S/MIME’. The impact of the security variability model sec-deltas on the corresponding security artefacts was then analyzed. The results of this analysis were the security application variability sub-model (partially shown in Fig. 5) along with the security requirements artefact deltas (assets, threats, etc.), which were developed and then correctly documented.

Finally, these sec-deltas were communicated to the security risk expert who estimated the risks of carrying out or not carrying out the security requirements deltas (task A2.4 “*Application Risk Assessment*”) as shown in Fig. 6. For example, the estimated security risk of email authenticity for not carrying out the security variant ‘S/MIME’ was ‘high’ (risk of 5 on a scale of 0–5) (named [BS_email] in Fig. 6). The first number of each cell in the table is the value of the assets. The second number of each cell is the degradation value of the assets caused by the threat expressed as a percentage. The third value is the accumulated impact on the assets, and the last value is the accumulated risk to the assets, according to the Magerit method. SREPPLineTool is additionally able to automatically calculate impacts and risks by introducing the degradation and likelihood of occurrence (an example of this is shown in Fig. 7).

The “*Application Security Requirements Negotiation and Prioritization*” task (task A2.5 of PLSecAppReq). After the application risk assessment of the sec-deltas was performed, the results were communicated to the security architect and to the security requirements engineer who estimated the realisation effort based on the sec-deltas and their associated risks. The stakeholders used this estimation to decide whether or not the security requirements deltas should be carried out and which security standard the application should fulfil. In this example we performed a slight economical analysis by balancing the risk with the economical impact of implementing countermeasures. We thereby reached an agreement with the stakeholders with regard to taking into account those security requirements associated with those threats that implied high or very high risk (risk of 4 or 5) whatever the conflicts with other requirements were. However, for the security requirements with a risk which was lower than high (that is, from 3 to 1, medium to low) we had to make trade-offs, mainly with other non-functional requirements, and particularly with regard to performance and interface accessibility. As Fig. 5 shows, if the ‘Internet web’ feature is selected, as occurs in this example, the e-bill service has to fulfil the WAI, Web Accessibility Initiative, level ‘AA’. The application security requirements and the corresponding security requirements artefacts and security application variability sub-model were defined as a result of this task.

In the “*Application Security Requirements Specification*” task (task A2.6 of PLSecAppReq) the application security artefacts, the sec-deltas and the traces between application security artefacts and the corresponding domain security artefacts were formally specified and documented. The security application variability model and the traceability links of the application security artefacts to the application-specific variability model were also documented, such as the security requirement specification in XML (according to the grammar specified in Appendix A) shown in Fig. 5. The estimated risk and realisation costs were even related to the sec-deltas in order to ensure that decisions regarding sec-deltas were traceable.

Finally, in task A2.7 (“*Application Security Requirements Inspection*”) the security requirements artefacts variability consistency between the application and domain artefacts of the e-billing reception service product line was verified. We also verified whether the security requirements satisfied the stakeholders' security needs and application security goals, and whether the security requirements conformed to ISO/IEC 27001 control objectives, to Common Criteria assurance requirements and to the IEEE 830-1998 standard. In addition, the product line manager decided to accept the suggestions for additional and altered security features made by the security requirements engineer. This signified that the security feature ‘S/MIME’ was included in the eBill-PL as an additional variant of the “Secure submissions” variation point along with the associated security requirements and their related security requirements artefacts (assets, threats, etc.) and their variability links.

Finally, among the most important lessons learnt from the case study presented above we can highlight the following:

- Tool support is critical for the practical application of this process to large-scale software systems owing to the number of artefacts handled and the complexity of the traceability relations and the variability model.
- We must improve the graphical interface of SREPPLineTool for the security variability definition to make this key task more intuitive for security requirements engineers who are not experts in SPL engineering. Moreover, integration with other tools of the SPL development paradigm is essential if an appropriate traceability of the security requirements artefacts and an appropriate implementation of the security requirements engineering are to exist in an organization.
- The following benefits have been obtained by the organization in which the case study was carried out: (1) it has managed to normalize a systematic and specific process for the management of security requirements in SPL which conforms to ISO/IEC 15408 and ISO/IEC 27001. (2) A security core assets repository has been created whose artefacts will be reused for the development of the products of the SPL and could also be reused for the development of future SPLs in the organization.
- We are now aware of SREPPLine's complexity and realise that it could be perfectly well applied in large companies or administrations. However, it needs to be adapted if it is to be applied in small and medium sized companies, because around three people are required to manage the process correctly.
- We are now aware that SREPPLine reduces the effort of the security requirements engineering phase in the development of more products in the same product line by around 50%. Furthermore, it ensures SPL certification against ISO/IEC 15408 and ISO/IEC 27001 with regards to security requirements issues, and it also reduces the effort of preparing ISO/IEC 15408 and ISO/IEC 27001 certification by around 30%.

5. Conclusions

Security requirements issues are extremely important in SPLs because a weakness in security can cause problems in all the products of an SPL. Although several attempts have been made to fill the gap between requirements engineering and SPL requirements engineering, no systematic approach with which to define security quality requirements and to manage their variability and their related security artefacts to the models of an SPL is available.

The contribution of this work is that of providing a systematic approach for the management of the security requirements and their variability from the early stages of product line development. The purpose of this is to facilitate the conformance of SPL products to the most relevant security standards with regard to the management of security requirements, such as ISO/IEC 27001 and ISO/IEC 15408 (Common Criteria). Our proposal defines an iterative, incremental and customizable process based on a Security Reference Meta Model driven by security standards in order to assist in SPL security requirements definition and to facilitate product security certification. This meta-model also assists in the management of the variability and traceability of the security requirements related artefacts and supports the capturing, specifying of and reasoning about security requirements and their artefacts. Our proposal consequently permits security variants to be selected in the requirements level rather than in the design level. It also provides a cross-cutting view of the security variability in all security development artefacts and assists in maintaining consistency in the different views of variable security requirements artefacts. Therefore, SREPPLine together with the Security Reference Meta Model is a particularly suitable approach for SPL in which security is a key issue. This is owing to the broader impact of the existence or non-existence of specific security features in all SPL members, and the level of management of variable security features required for the diversity of market segments. Our approach is also scalable thanks to the fact that SREPPLine is an add-in of tasks. Since not all the steps of each task are required, developers could create their own lightweight process by selecting a subset of the steps of each task in order to adapt the process to the size of the project and the organization.

In contrast with our previous works, in this paper a detailed explanation is provided, and the SREPPLine process is formally specified with SPEM 2.0 [51], specifying: roles, input & output artefacts, activities, tasks & steps. We also define a grammar in XML for the Security Reference Meta Model. By using these standards SREPPLine better facilitates integration with other processes and repositories. Furthermore, we present a framework in which these components will be integrated with the aim of providing a holistic security requirements framework for the development of secure SPLs.

Finally, in the future we plan to work to facilitate the use of other risk assessment methods which conform to ISO/IEC 27005 in SREPPLine, such as OCTAVE [2]. We shall extend our proposed process to other phases of product line development. We shall also develop a version of SREPPLine which has been customized for small and medium companies, and which is less complex and will therefore require less effort and money for its application. Further work is also required to refine the prototype of SREPPLineTool that we have developed to support SREPPLine and the Security Resources Repository, which was one of the lessons learned in the case study performed at the Spanish Public Administration partially described in [45]. This will be done in order to assist in the complex management and maintainability of the Security Reference Meta Model and its variability and traceability relations. The process, the meta-model and the tool have been shown to work well in a small SPL with relatively few artefacts and therefore few security requirements related artefacts, as in [45] and in the example briefly described in this paper. They have not yet, however, been tested in large product lines. We shall therefore carry out a refinement of our process, the tool and the meta model by testing them with a complete and exhaustive real industrial case study in a large SPL in order to validate and illustrate SREPPLine in far greater depth. We thus aim to provide a holistic and validated framework for security requirements engineering in SPL.

Acknowledgments

This research is part of the BUSINESS (PET2008-0136) Project of the Ministry of Science and Innovation (Spain), PEGASO/MAGO (TIN2009-13718-C02-01) Project of the Ministry of Science and Innovation (Spain) and the FEDER. Moreover, it is part of the SISTEMAS (PII2I09-0150-3135) and QUASIMODO (PAC08-0157-0668) Projects of the Castilla – La Mancha Regional Government and the FEDER as well as it is part of the MEDUSAS (IDI-20090557) Project of the Ministry of Science and Innovation (Spain- CDTI).

Appendix A. Grammar of the security reference model

This appendix provides details of the grammar for the different types of artefacts considered in the Security Reference Model: security feature, asset, threat and security requirement along with their respective categories.

In Fig. 8 "$*$" represents multiplicity; "{$\cdots$}" represents additional information that could be filled in; and the artefacts are between "<$\cdots$>"). The other elements are depicted in the figures (Figs. 9–12). This grammar registers the elements and their relations which are managed by the meta model. As shown in Fig. 8, the Security Reference Model grammar is based on five core elements: the security reference model, the variability element,

```
security reference model::=
    <sec-ref model>
      <variability element>*
    </sec-ref model>

variability element::=
  <variability element>
    <variability_element_id>
    {variability variant variation-point}
    #name#
    [description]
    <dependence>*
    <var-dependence>*
  </variability element>

dependence::=
  <dependence>
    #name#
    {artefact_id  artefact_name
      artefact_category}*
  </dependence>
```

```
var-dependence::=
  <var-dependence>
    #name#
    {var-dependence-type optional
      mandatory}
    {variability element}*
  </var-dependence>

artefact::=
  <artefact>
    #name#
    {artefact-type application product-
    line}
    <security_feature asset
      security_objective threat
      security_requirement>
  </artefact>
```

Fig. 8. Core of the Grammar of the Security Reference Model.

```
security feature categories::=
  <security feature categories>
    {security feature category}*
  </security feature categories>

security feature category::=
  <security feature category>
    <security feature id>
    #category name#
    [description] [asset categories]
    [Common Criteria classes]
    [ISO27001 clauses]
    {security feature category}*
  </security feature category>

security feature::=
  <security feature>
    <security feature id>
    #security feature name#
    [description]
    <security feature categories>
    <dependence>*
  </security feature >

security feature app::=
  <security feature app>
    <security feature app id>
    #security feature app name#
    [description]
    <security feature id> (security
      feature of the SPL domain)
    <security feature categories>
    <dependence>*
  </security feature app>
```

```
asset categories::=
  <asset categories>
    {asset category}*
  </asset categories>

asset category::=
  <asset category>
    <category id>
    #category name#
    [description]
    [security objective categories]
    [security feature categories]
    {asset category}*
  </asset category>

asset::=
  <asset>
    <asset id>
    #asset name#
    [description]
    <asset categories>
    <dependence>*
  </asset>

asset-app::=
  <asset-app>
    <asset-app id>
    #asset-app name#
    [description]
    <asset id>  (asset del dominio LPS)
    <asset categories>
    <dependence>*
  </asset-app>
```

Fig. 9. Grammar of the Security Reference Model (security features and assets).

dependence, var-dependence and artefact. After Fig. 8 it is provided details of the grammar used for the different types of artefacts that are considered in the Security Reference Model: security feature, asset, threat and security requirement, along with their respective categories.

As is described in Fig. 8, a ‘security reference model’ is composed of variability elements. A ‘variability element’ is a variant or a variation point, which could have variability dependences (‘var-dependence’) with other ‘variability elements’, that is, with other variants or variation points. The grammar therefore registers

```
security objectives categories::=
  <security objectives categories>
    {security objective category}*
  </security objectives categories>

security objective category::=
  <security objective category>
    <category id>
    #category name#
    [description] [threat
categories]
    [asset categories]
  </security objective category>

security objective::=
  <security objective>
    <security objective id>
    #security objective name#
    [description]
    < <asset> <security objective>
      <value> >*
    <security objective category>
    <dependence>*
  </security objective >
```

```
security objective app::=
  <security objective app>
    <security objective app id>
    #security objective app name#
    [description]
    < <asset-app> <security objective app>
      <value> >*
    <security objective id>  (SPL domain
    security objective)
    <security objective category>
    <dependence>*
  </security objective app>
```

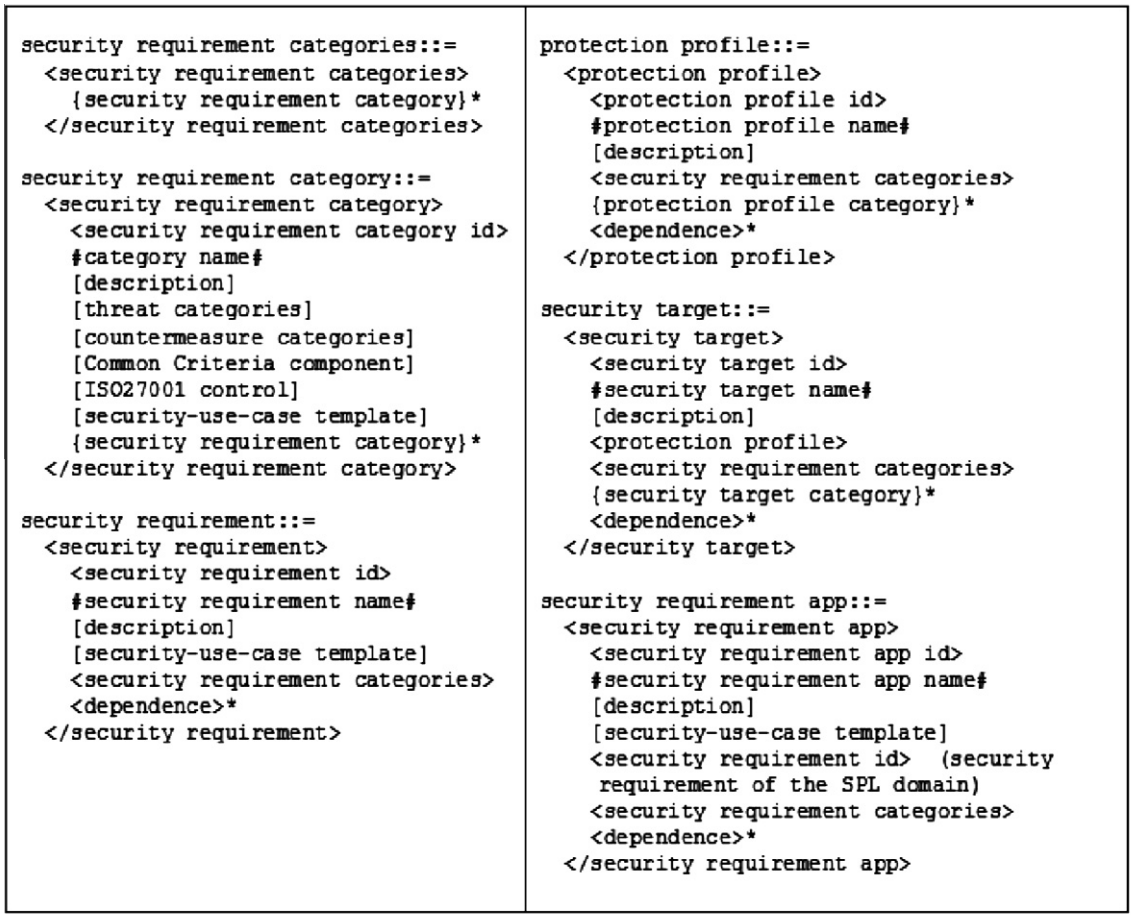
Fig. 10. Grammar of the Security Reference Model (security objectives).

```
threat categories::=
  <threat categories>
    {threat category}*
  </threat categories>

threat category::=
  <threat category>
    <category id>
    #category name#
    [description]
    [security objective categories]
    [asset categories]
    [security requirement
categories]
    [Common Criteria families]
    [ISO27001 control objectives]
    [misuse-case template]
    [attack-tree template]
    {threat category}*
  </threat category>
```

```
threat::=
  <threat>
    <threat id>
    #threat name#
    [description]
    < <asset>, <security objective>
       <degradation_value> <frequency_value>
       <impact_value> <risk_value>
       [<residual_risk_value>] >*
    [misuse-case template]
    [attack-tree template]
    <threat categories>
    <dependence>*
  </threat>

threat-app::=
  <threat-app>
    <threat-app id>
    #threat-app name#
    [description]
    < <asset-app>, <security objective>
       <degradation_value> <frequency_value>
       <impact_value> <risk_value>
       [<residual_risk_value>] >*
    <threat id>  (threat of SPL domain)
    [misuse-case template]
    [attack-tree template]
    <threat categories>
    <dependence>*
  </threat-app>
```

Fig. 11. Grammar of the Security Reference Model (threats).

the 'variability dependency'. A 'variability element' could also have 'dependences' with 'artefacts', and this is how the grammar registers the 'VP artefact dependency' and 'artefact dependency' depicted in Fig. 2. An 'artefact' may be part of the application or of



```
security requirement categories::=
  <security requirement categories>
    {security requirement category}*
  </security requirement categories>

security requirement category::=
  <security requirement category>
    <security requirement category id>
    #category name#
    [description]
    [threat categories]
    [countermeasure categories]
    [Common Criteria component]
    [ISO27001 control]
    [security-use-case template]
    {security requirement category}*
  </security requirement category>

security requirement::=
  <security requirement>
    <security requirement id>
    #security requirement name#
    [description]
    [security-use-case template]
    <security requirement categories>
    <dependence>*
  </security requirement>
```

```
protection profile::=
  <protection profile>
    <protection profile id>
    #protection profile name#
    [description]
    <security requirement categories>
    {protection profile category}*
    <dependence>*
  </protection profile>

security target::=
  <security target>
    <security target id>
    #security target name#
    [description]
    <protection profile>
    <security requirement categories>
    {security target category}*
    <dependence>*
  </security target>

security requirement app::=
  <security requirement app>
    <security requirement app id>
    #security requirement app name#
    [description]
    [security-use-case template]
    <security requirement id>  (security
     requirement of the SPL domain)
    <security requirement categories>
    <dependence>*
  </security requirement app>
```

Fig. 12. Grammar of the Security Reference Model (security requirements).

the product line and must be one of the following types (whose grammars are shown next): security feature, asset, threat and security requirement.

References

[1] S. Abu-Nimeh, S. Miyazaki, N.R. Mead, Integrating privacy requirements into security requirements engineering, in SEKE, 2009, pp. 542–547.

[2] C. Alberts, A. Dorofee, OCTAVE Method Implementation Guide v2.0, C.M.U. Software Engineering Institute, Editor, 2001, Pittsburgh (USA).

[3] J.L. Arciniegas, J.C. Dueñas, J.L. Ruiz, R. Cerón, J. Bermejo, M.A. Oltra, Architecture reasoning for supporting product line evolution: an example on security, in: T. Käkölä, J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management, Springer, 2006.

[4] L. Aversano, G. Canfora, A.D. Lucia, P. Gallucci, Business process reengineering and workflow automation: a technology transfer experience, The Journal of Systems and Software (2002) 29–44.

[5] L. Baresi, S. Morasca, Three empirical studies on estimating the design effort of web applications, ACM Transactions on Software Engineering and Methodology (TOSEM) 16 (4) (2007) 15-1–15-40.

[6] Bastian Best, Jan Jürjens, B. Nuseibeh, Model-based security engineering of distributed information systems using UMLsec, in: 29th International Conference on Software Engineering (ICSE 2007), 2007, pp. 581–590.

[7] J. Bayer, S. Gerard, O. Haugen, J. Mansell, B. Moller-Pedersen, J. Oldevik, P. Tessier, J.-P. Thibault, T. Widen, Consolidated product line variability modeling, in: T.Käkölä, J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management, 2005, pp. 195–241.

[8] P. Berander, A. Andrews, Requirements prioritization, in: A. Aurum, C. Wohlin (Eds.), Engineering and Managing Software Requirements, 2005, pp. 69–94.

[9] A. Birk, G. Heller, Challenges for requirements engineering and management in software product line development, in: International Conference on Requirements Engineering (REFSQ 2007), 2007, pp. 300–305.

[10] J. Bosh, Design & Use of Software Architectures, Pearson Education Limited, 2000.

[11] S. Bühne, G. Halmans, K. Lauenroth, K. Pohl, Scenario-based application requirements engineering, in: T. Käkölä, J.C. Dueñas (Eds.), Software Product Lines – Research Issues in Engineering and Management, 2005, pp. 161–194.

[12] CERT/CC, CERT/CC Statistics 1995–2008, 2009. <http://www.cert.org/stats/fullstats.html>.

[13] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns, SEI Series in Software Engineering, Addison-Wesley, 2002.

[14] L. Compagna, P.E. Khoury, A. Krausová, F. Massacci, N. Zannone, How to integrate legal requirements into a requirements engineering methodology for the development of security and privacy patterns, Artificial Intelligence and Law 17 (1) (2009) 1–30.

[15] K.-K.R. Choo, R.G. Smith, R. McCusker, Future directions in technology-enabled crime: 2007–09, in: Research and Public Policy Series, Australian_Government, Editor, 2007, Australian Institute of Criminology.

[16] T.E. Faegri, S. Hallsteinsen, A software product line reference architecture for security, in: T. Käkölä, J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management, Springer, 2006.

[17] D.G. Firesmith, Engineering security requirements, Journal of Object Technology 2 (1) (2003) 53–68.

[18] D.G. Firesmith, Security use cases, Journal of Object Technology (2003) 53–64.

[19] D.G. Firesmith, Engineering safety and security related requirements for software intensive systems, in: International Conference on Software Engineering, IEEE Computer Society, 2007, p. 169.

[20] P. Giorgini, H. Mouratidis, N. Zannone, Modelling security and trust with Secure Tropos, in: H. Mouratidis, P. Giorgini (Eds.), Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing, 2007, pp. 160–189.

[21] P. Grünbacher, N. Seyff, Requirements negotiation, in: A. Aurum, C. Wohlin (Eds.), Engineering and Managing Software Requirements, 2005, pp. 143–162.

[22] C.B. Haley, R. Laney, J.D. Moffet, B. Nuseibeh, Security requirements engineering: a framework for representation and analysis, IEEE Transactions on Software Engineering 34 (1) (2008) 133–153.

[23] ISO/IEC, ISO/IEC 13335 Information Technology – Security Techniques – Management of Information and Communications Technology Security, 2004.

[24] ISO/IEC, ISO/IEC 15446 Information Technology – Security Techniques – Guide for the Production of Protection Profiles and Security Targets, 2004.

[25] ISO/IEC, ISO/IEC 15408:2005 Information Technology – Security Techniques – Evaluation Criteria for IT Security (Common Criteria v3.0), 2005.

[26] ISO/IEC, ISO/IEC 27001 Information Technology – Security Techniques – Information Security Management Systems – Requirements, 2006.

[27] J. Jürjens, UMLsec: extending UML for secure systems development, in: UML 2002 – The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference, LNCS 2460, 2002, pp. 412–425.

[28] J. Jürjens, Secure Systems Development with UML, Springer Academic Publishers, 2005.

[29] J. Jürjens, J. Schreck, Y. Yu, Automated analysis of permission-based security using UMLsec, in: Fundamental Approaches to Software Engineering (FASE 2008), held as part of the Joint European Conferences on Theory and Practice of Software (ETAPS 2008), 2008, pp. 292–295.

[30] K. Kang, S. Cohen, J.A. Hess, W.E. Novak, S.A. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Software Engineering Institute, Carnegie-Mellon University, 1990.

[31] H.-K. Kim., Automatic Translation Form Requirements Model into Use Cases Modeling on UML. ICCSA 2005, LNCS, 2005, pp. 769–777.

[32] C. Kuloor, A. Eberlein, Aspect-oriented requirements engineering for software product lines, in: Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03), 2003.

[33] A.v. Lamsweerde, Elaborating security requirements by construction of intentional anti-models, in: 26th International Conference on Software Engineering (ICSE 2004), 2004, pp. 148–157.

[34] J. Lasheras, R. Valencia-García, J.T. Fernández-Breis, A. Toval, An ontology-based framework for modelling security requirements, in: The 6th International Workshop on Security in Information Systems – WOSIS, 2008.

[35] H.A. Linstone, M. Turoff, in: H.A. Linstone, M. Turoff (Eds.), The Delphi Method: Techniques and Applications, Addison-Wesley Publishing Company, 1975.

[36] L. Liu, E.S.K. Yu, J. Mylopoulos, Security and privacy requirements analysis within social setting, in: 11th IEEE International Requirements Engineering Conference (RE'03), 2003.

[37] MAP, Methodology for Information Systems Risk Analysis and Management, in: Ministry for Public Administration, 2005.

[38] F. Massacci, M. Prest, N. Zannone, Using a security requirements engineering methodology in practice: the compliance with the Italian data protection legislation, Computers Standards and Interfaces (2005) 445–455.

[39] N.R. Mead, E.D. Hough, Security requirements engineering for software systems: case studies in support of software engineering education, in: CSEE&T, 2006, pp. 149–158.

[40] N.R. Mead, E. Hough, T. Stehney, Security Quality Requirements Engineering (SQUARE) Methodology, (CMU/SEI-2005-TR-009), Software Engineering Institute, Carnegie Mellon University, Pittsburgh (USA), 2005.

[41] D. Mellado, E. Fernández-Medina, M. Piattini, A comparative study of proposals for establishing security requirements for the development of secure information systems, in: The 2006 International Conference on Computational Science and its Applications (ICCSA 2006), vol. 3, Springer LNCS 3982, 2006, pp. 1044–1053.

[42] D. Mellado, E. Fernández-Medina, M. Piattini, A common criteria based security requirements engineering process for the development of secure information systems, Computer Standards and Interfaces 29 (2) (2007) 244–253.

[43] D. Mellado, E. Fernández-Medina, M. Piattini, Security requirements variability for software product lines, in: Symposium on Requirements Engineering for Information Security (SREIS 2008) co-located with ARES 2008, 2008, pp. 1413–1420.

[44] D. Mellado, C. Blanco, L.E. Sanchez, E. Fernández-Medina, A systematic review of security requirements engineering, Computers Standards & Interfaces 32 (2010) 153–165.

[45] D. Mellado, E. Fernández-Medina, M. Piattini, Towards security requirements management for software product lines: a security domain requirements engineering process, Computer Standards & Interfaces 30 (2008) 361–371.

[46] D. Mellado, E. Fernández-Medina, M. Piattini, A systematic review of security requirements engineering, Computers Standards & Interfaces, 2010. <http://dx.doi.org/10.1016/j.csi.2010.01.006> (accessed 02.02.10).

[47] D. Mellado, J. Rodríguez, E. Fernández-Medina, M. Piattini, Automated support for security requirements engineering in software product line domain engineering, in: The Fourth International Conference on Availability, Reliability and Security (ARES 2009), 2009.

[48] E. Niemelä, A. Immonen, Capturing quality requirements of product family architecture, Information & Software Technology (2007) 1107–1120.

[49] OECD, The promotion of a culture of security for information systems and networks in OECD countries, in: DSTI/ICCP/REG(2005)1/FINAL, Organisation for Economic Co-operation and Development, 2005.

[50] OMG, Reusable Assets Specification (RAS), 2004 (ptc/04-06-06).

[51] OMG, Software & Systems Process Engineering Meta-Model Specification v.2.0, 2008. <http://www.omg.org/spec/SPEM>.

[52] A.L. Opdahl, G. Sindre, Experimental comparison of attack trees and misuse cases for security threat identification, Information and Software Technology 51 (5) (2010) 916–932.

[53] K. Pohl, G. Böckle, F.v.d. Linden, Software Product Line Engineering, Foundations, Principles and Techniques, Berlin Heidelberg, Berlin, 2005.

[54] K. Schmid, I. John, A customizable approach to full-life cycle variability management, Science of Computer Programming, vol. 53, Elsevier, 2004, pp. 259–284.

[55] K. Schmid, K. Krennrich, M. Eisenbarth, Requirements Management for Product Lines: A Prototype, Fraunhofer IESE, 2005.

[56] B. Schneier, Attack trees, Dr. Dobb's Journal 24 (12) (1999).

[57] SEI, +SAFE, V1.2 A Safety Extension to CMMI-DEV V1.2. 2007, Pittsburgh (USA): Software Engineering Institute, Carnegie Mellon University.

[58] G. Sindre, D.G. Firesmith, A.L. Opdahl. A reuse-based approach to determining security requirements, in: Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Austria, 2003.

[59] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, Requirements Engineering 10 (1) (2005) 34–44.

[60] M. Sinnema, S. Deelstra, J. Nijhuis, J. Bosch, COVAMOF: a framework for modeling variability in software product families, in: Proc. of the Third Softw. Product Line Conf. (SPLC 2004), Boston, MA, USA, 2004.

[61] A. Toval, J. Nicolás, B. Moros, F. García, Requirements reuse for improving information systems security: a practitioner's approach, Requirements Engineering (2002) 205–219.

[62] E. Yu, L. Liu, Mylopoulos, A Social Ontology for Integrating Security and Software Engineering, in: Integrating Security and Software Engineering: Advances and Future Visions, Idea Group Publishing, 2007.