Secure Tropos Framework for Software Product Lines Requirements Engineering

Daniel Mellado[1], Haralambos Mouratidis[2] and Eduardo Fernández-Medina[3]

[1] *Spanish Tax Agency, Large Taxpayers Department, IT Auditing Unit.*
Paseo de la Castellana 106, 28046 Madrid (Spain).
damefe@esdebian.org

[2] *School of Architecture, Computing and Engineering, University of East London.*
4-6 University Way, Docklands; E16 2RD, London (U.K.).
haris@uel.ac.uk

[3]*GSyA Research Group, University of Castilla-La Mancha, Information Systems and Technologies Department.*
Paseo de la Universidad 4, 13071 Ciudad Real (Spain).
Eduardo.FdezMedina@uclm.es

Abstract

Security and requirements engineering are two of the most important factors of success in the development of a software product line (SPL) due to the complexity and extensive nature of them, given that a weakness in security can cause problems throughout the products of a product line. Goal-driven security requirements engineering approaches, such as Secure Tropos, have been proposed in the literature as a suitable paradigm for elicitation of security requirements and their analysis on both a social and a technical dimension. Nevertheless, on one hand, goal-driven security requirements engineering methodologies are not appropriately tailored to the specific demands of SPL, while on the other hand specific proposals of SPL engineering have traditionally ignored security requirements. This paper presents work that fills this gap by proposing "SecureTropos-SPL" framework, an extension to Secure Tropos to support SPL security requirements engineering which is based on security goals and driven by security risks.

Keywords*: Security requirements, product lines, requirements engineering, security requirement engineering, Secure Tropos.*

1. Introduction

Information systems undoubtedly play an important role in today's society and more and more are at the heart of critical infrastructures. It is widely accepted in the security research literature [14], that security is of particular importance to such information systems and that is essential for security to be considered from the early stages of software development for an effective management of security issues. Although security is traditionally considered a technical issue; security is, in fact, a two-dimensional problem, which involves technical as well as social challenges [18].

At the same time, in recent years, many public and private organizations are making the strategic decision to adopt a software product line (SPL) approach to the production of software-intensive systems [13]. Since SPL strategy has proven successful at reducing both time-to-market and development costs [4, 6] and obtaining both high-quality information systems and higher productivity [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.

Proper analysis and understanding of security requirements are important because they help us to discover any security or requirement defects or mistakes in the early stages of development, in fact the long-standing credo of requirements engineering reads: ‘‘If you don’t know what you want, it’s hard to do it right’’ [7]. In SPL development it is even more important given that a weakness in security owing to a mistake in a security requirement can cause problems throughout the products of a product line. Therefore, the elicitation of security requirements for SPL is a challenging task, mainly due to 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.

Nevertheless, there is lack of approaches in the security requirements literature [14], which would support the elicitation and analysis of both social and technical security requirements from the early stages of the SPL development process. On one hand current SPL approaches which include partial support for security requirements engineering do not manage both dimensions of security (social and technical dimension); on the other hand, proposals that manage both the technical and the social dimensions of security (such as Secure Tropos) are not tailored enough to support the SPL development paradigm.

In this paper, we propose *SecureTropos-SPL*, an extension of some stages of Secure Tropos [17] methodology to fill this gap. Our work initially aligns SPL concepts to Secure Tropos concepts, and secondly it redefines the Secure Tropos process, so that we proposed a risk-driven goal-based process to manage security requirements variability at both Early Requirements and Late Requirements stages of Secure Tropos in SPL development. Finally, it is proposed the extension of Secure Tropos metamodel and language to support security risks and SPL concepts such as ‘variability’ and its modeling, that is SPL modeling with Secure Tropos, in order to manage at the same time both the technical and the social dimensions of SPL security and also taking into account the security risks.

This paper is structured as follows. Section 2 describes the background information about Secure Tropos and SPL needed for a better understanding of the proposal. In Section 3 it is sum up the related work. Section 4 outlines the core elements of *SecureTropos-SPL*, our proposed extensions to Secure Tropos, while Section 5 illustrates with the aid of an example the applicability of these extensions to Secure Tropos. Finally, Section 6 discusses contributions and future work.

2. Secure Tropos and Software Product Lines Requirements Engineering Basics

2.1. Overview of Secure Tropos

Secure Tropos [17] is a security-oriented extension of the widely known requirements engineering methodology Tropos [5]. It introduces a number of security-related concepts to the Tropos methodology. Tropos (and as a result Secure Tropos) methodology is mainly based on four stages:

- *Early requirements* analysis aimed at defining and understanding a problem by studying its existing organizational setting.
- *Late requirements* analysis conceived to define the system-to-be in the context of its operational environment.
- *Architectural design*, that deals with the definition of the system global architecture in terms of subsystems; and the
- *Detailed design* phase, aimed at specifying each architectural component in further detail, in terms of inputs, outputs, control and other relevant information.

The main unique points of the methodology compared to other security oriented software engineering approaches are that

- social issues of security are analyzed during the early requirements stage;
- security is considered simultaneously with the other requirements of the system-to-be; and
- the methodology supports not only requirements stages but also design stages.

In this paper we will extend Secure Tropos in order to to manage security requirements variability at both Early Requirements and Late Requirements stages of Secure Tropos in SPL development

2.2. Software Product Lines Requirements Engineering Basics

A software product line (SPL) is a set of software-intensive systems sharing a common, managed set of features [10] 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 [6]. 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 [23]. 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 customised 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 decide 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 [21]. Domain engineering is the process of SPL engineering in which commonality and variability of the product line are defined and carried out. According to [21] 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 [6].

3. Related Work

Several attempts have also recently been made to define SPL architectures for security, such as the approach of Faegri et al. [8] and the approach of Arciniegas et al. [1], although their work is focused on tackling security management in SPL engineering, their approach is applicable to the latest stages of the development process rather than security requirements, because are more orientated towards the software solution than to security requirements elicitation and definition or include only a few security requirements tasks, but without managing all the security requirements artefacts (assets, threats, etc.). The Security Requirements Engineering Process for Software Product Lines (SREPPLine) [15] has been recently proposed to support security requirements analysis for SPL. However, SREPPLine fails to consider both the social and technical dimensions of security and it also does not support the parallel modelling of security requirements and the rest of the security elements and their variability with a homogeneous modelling language, as our approach does.

The most relevant “generic” security requirements related proposals were systematically reviewed in [14] (Secure Tropos included). Thanks to this review that we have already done, it can be observed that 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. Moreover, they do not provide a methodological tailored approach for SPL engineering, that is, they do not have specific activities nor language to manage the security variability needed by the SPL development paradigm. Therefore, they are not appropriate enough to manage security requirements in SPL, as it was also explained in [15].

Having said this, each of these approaches makes highly important contributions to security requirements engineering in SPL. In addition, some of their features are used as the basis of our proposal.

4. SecureTropos-SPL: Secure Tropos Framework for Software Product Lines

In this section, we present the major principles of our proposal. Firstly, we outline the core of our approach. Next we align SPL concepts to Secure Tropos concepts, and then it is redefined the Secure Tropos process at both Early Requirements and Late Requirements phases of Secure Tropos. Finally, it is proposed an extension of Secure Tropos language in order to deal with the variability needed for SPL engineering and to manage security risks elements.

4.1. Overview of our approach

Our approach 'Secure Tropos – SPL', as shown in Fig. 1, is based on Secure Tropos and therefore on Tropos methodology. We propose an extension to Secure Tropos to support SPL security requirements engineering based on security goals and driven by security risks.

The aim of our approach is to minimize both knowledge of the necessary security and risks concepts and security expert participation during SPL product development, so that this approach provides support for the elicitation and analysis of both social and technical security requirements following security risks criterion from the early stages of the SPL development process.

Consequently, we aligned concepts of Goal Driven SPL Engineering with Secure Tropos concepts. Furthermore, we have redefined Secure Tropos process by means of introducing new tasks which deal with the variability of the security requirements, as well as we have specified these new tasks and activities of the process using the OMG standard SPEM 2.0 [20]. As shown in Fig. 1, we have integrated the *Domain* and *Application Requirements Engineering* activities in both the *Early* and *Late Requirements* stages, as well as we suggested a new task '*Variability Analysis*' which is carried out during *Domain Requirements Engineering Activity* in order to manage the variability of SPL, so that the common and variable security goals are identified and modelled. We have also proposed two new tasks: '*Variability Instantiation*' and '*Sec-deltas Analysis*', which are performed in *Application Requirements Engineering Activity* and in these tasks the set of domain security goals are instantiated as well as it is analyzed and modelled the security specific requirements of the application. Moreover, we introduced security risk related tasks based on security requirements management approaches (such as Magerit [12], methodology officially recognised by NATO at the 9th NATO cyberdefense workshop in 2008 and by OECD [19]) with the aim of introducing risk criterion in the security requirements elicitation, so that our approach not only consider both the social and technical dimensions of security it also does take into account the security risk in SPL engineering. Moreover, our final aim is also to align the work with relevant "industrial" standards or guidelines (as for example Magerit, …) and methods to drive the use of the work in industry.

In Fig. 1 we have outlined the main components of our proposed framework as a high level abstraction diagram of components. In the left side of the figure, it is depicted how our framework fits in the SPL development paradigm and in the requirements engineering methodology Tropos [5]. In the top of the figure, we have sum up the key languages, tools or techniques that our framework is based

on, such as SPEM 2.0 to specify the process, SecTroModellingTool to model according Secure Tropos specification, and security risk assessment approaches (as Magerit, CRAMM or Octave, etc.). In the bottom of the figure, it is shown the extension of Secure Tropos metamodel and language as it is explained next as a subsection, so that we have added new entities and relationships to support variability and risk elements in two sub-parts of the Secure Tropos modetamodel related to the Security Enhanced Actor Model (SEAM) and Security Enhanced Goal Model (SEGM). Finally, in the centre of the figure, we represent the core activities of the process proposed in our framework.

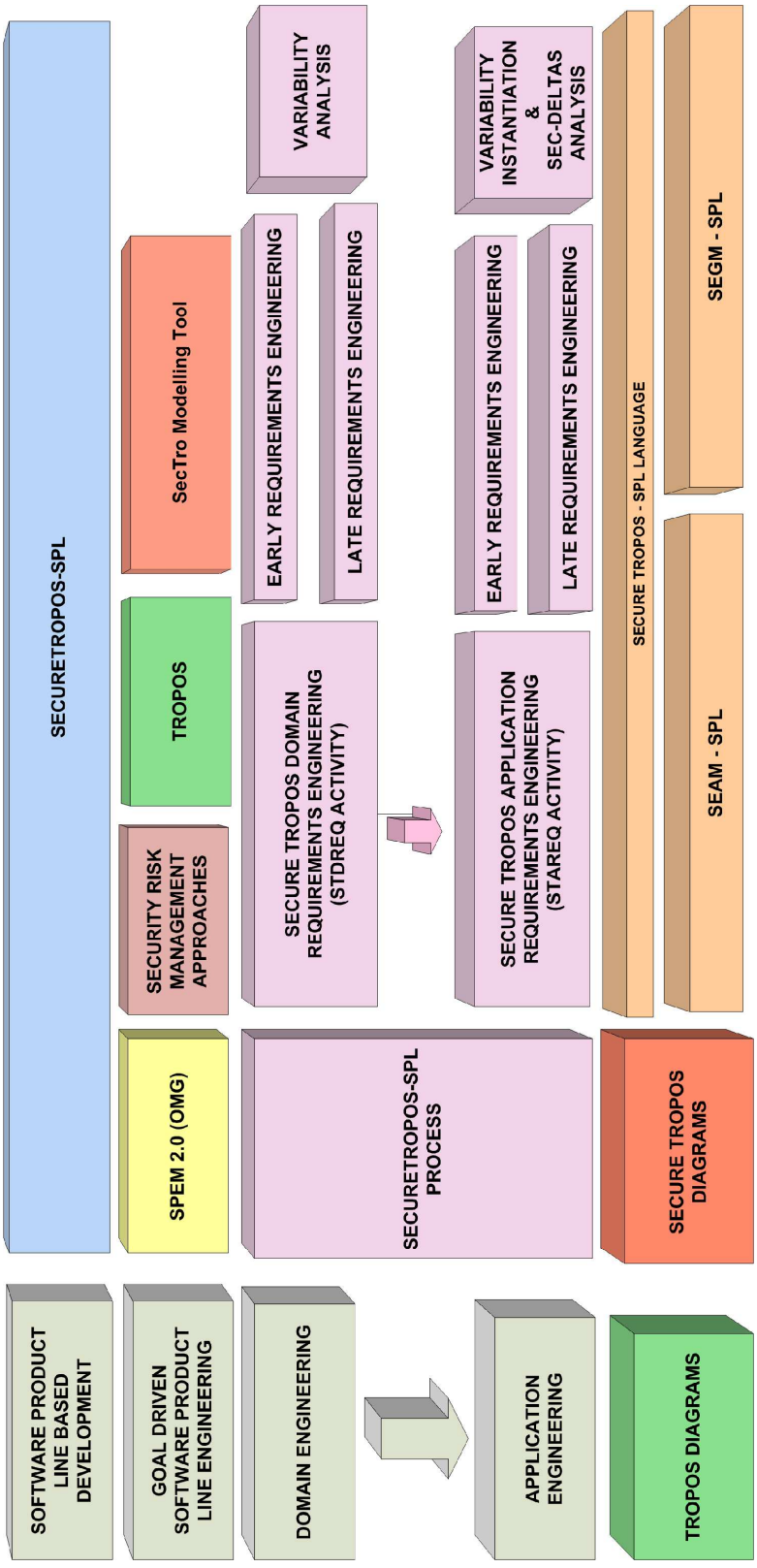

Fig. 1 Secure Tropos - SPL overview

4.2. Aligning Secure Tropos with SPL concepts

One of the first challenges we faced, was the alignment between Secure Tropos concepts and SPL concepts. Firstly, we had to introduce the concept of variability

in Secure Tropos, due to the fact that variability management is at the heart of the SPL paradigm.

Secure Tropos is a goal driven security requirements engineering methodology, in which a *goal* represents actors' strategic interests and a *secure goal* represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced to achieve possible security constraints that are imposed to an actor or exist in the system. An *actor* is defined as an entity that has strategic goal. In Secure Tropos *security constraints* define the system's security requirements; they are security conditions imposed to an actor that restricts achievement of an actor's goals, execution of plans or availability of resources. In addition, Secure Tropos defines secure dependencies. A *secure dependency* introduces security constraint(s) that must be fulfilled for the dependency to be satisfied.

In SPL engineering (but above all in goal driven SPL engineering), since a goal could provide the rationale for variations in domain requirements [11], we used it as a discriminator that enables us to identify common and variant goals and hence secure goals in Secure Tropos. Thus, the common (default option), optional and/or alternative goals in SPL can be modeled in Secure Tropos by means of a *Variability Dependency relationship* (a new relationship of Secure Tropos explained in next subsection 4.3 and shown in [16]).

Moreover, in Secure Tropos, the precise definition of how a secure goal can be achieved is given by a *secure plan*, which is defined as a particular way for satisfying a secure goal. Usually, a secure plan or goal needs a *secure resource*, which is an informational entity that is needed for the achievement of a secure goal or the fulfilment of a secure plan. Therefore, these entities of Secure Tropos (security constraint, secure plan, secure resource) could be part of variants of a SPL because they are related to goals and secure goals which are variations of a SPL, so that they are modeled by means of a variability dependency relationship between them and an actor, by means of the '*Variation'* entity (a new entity of Secure Tropos explained in next subsection 4.3 and shown in[16]).

We have also had to partially adapt the concept of *actor* of Secure Tropos, so that a SPL is a special type of a general actor and so as to during Application Engineering in SPL the different products/applications instantiated from the SPL are modeled as actors that reuse from the SPL-actor the domain common requirements, but also each application of the SPL will model their security specific requirements (security requirements are security related restrictions to the functionalities of the system) or each application will model the sec-deltas [15]. Sec-deltas occur when stakeholder security requirements cannot be completely satisfied by security domain requirements artefacts.

A second challenge was to integrate the two main activities related to requirements engineering in SPL engineering with Secure Tropos process (which is more detailed in next subsection 4.4). According to the definitions of these activities (previously explained in Section 2), and taking into account the development stages of Secure Tropos, we have integrated the *Domain* and *Application Requirements Engineering* activities in both the Early and Late requirements stages, although in *Application Requirements Engineering activity* during the Early Requirements stage it will only be done the inheritance of the common requirements of the SPL. That is, for the development of a SPL during the *Domain Requirements Engineering* activity we will carry out *Early* and *Late requirements stages* analyses, initially by defining and understanding the SPL

settings and then by defining the SPL-to-be in the context of its operational environment (modeling common, alternative and optional entities). While for the instantiation of the products/applications of the SPL during the *Application Requirements Engineering* activity we will inherit the early requirements from the SPL. Thus, during *Application Engineering* it will only be needed to carry out the *Late Requirements Engineering* stage of Secure Tropos, because is in this stage when each instantiated product/application from the SPL is defined, in the context of its operational environment, and when sec-deltas will be modeled.

Therefore, through the above discussed alignments and adaptations of concepts as well as the extensions of part of the Secure Tropos metamodel related to. Security Enhanced Actor Diagram (SEAD) and Security Enhanced Goal Diagram (SEGD) (explained in more detail in following subsection 4.3), we are able to capture and model security, with Secure Tropos, the security requirements of a SPL along with the variability of their related entities.

4.3. Secure Tropos Metamodel and Language Extension

Most existing variability management approaches in SPL, such as, for example: [2, 22, 24] are focused on addressing functional requirements variability and they do not manage the technical and the social dimensions of security of SPL. Hence, in this work with the aim of filling this gap, we are interested in two sub-parts of the Secure Tropos metamodel related to the Security Enhanced Actor Model (SEAM) and Security Enhanced Goal Model (SEGM), so that we have extended these parts of the metamodel in order to provide support to the variability management (which is the core of SPL engineering) in Secure Tropos metamodel as well as to the security risk assessment from the early stages of SPL development.

The SEAM defines a set of actors along with their secure dependencies and any security constraints that might be imposed to these actors. The SEGM assists to analyse the security issues of a particular Actor by understanding the implications that Security Constraints, identified in SEAM, have in that particular actor.

The extension to SEAM is shown in Fig. 2. We have added the '*Variability Dependency'* relationship, which inherits from '*Dependency'* and from which '*Secure Dependency*' inherits, so that variability of *Dependum* entities could be modelled. Furthermore, through the attribute 'Depender' or 'Dependee', developers can specify the "owner" of the variant. Secure Dependency relationships are 'Common' variants by default. Hence, through this new '*Variability Dependency'* relationship it is possible to state variability dependencies between all the entities supported by Tropos (i.e. goals, plans, actors, resources, and security constraints) and specify if they are 'common' (default), 'optional' or 'alternative' variability relationships. Nevertheless, the variability of the entities will start from the identification and specification of the goals variability, as it is the core of the variability because our framework is a goal-driven one.

Furthermore, we have inserted the '*Asset value*' as an attribute of the '*Security Constraint*', '*Plan', 'Goal'* and *'Resource'* elements in order to record the value

of the business and system / SPL assets[1] following a standardized scale from 0 to 10 in accordance with the Magerit [12] risk assessment methodology and agreed with the stakeholders. Through this new attribute '*Asset value*' in the '*Secure'* elements it is possible that each asset has a 'value' according to his related goal and/or secure goal. We based our asset analysis on the definition of an *asset* as anything that has value to the organization [9], that is, these assets are the resources in the information systems of the SPL, or these which are related to them which are necessary for the organization to operate correctly and to achieve its goals (both tangible or intangible). Thus, as we had identified that in Secure Tropos the entities: *plan*, *resource*, *goal* and *actor*; are used to model both business and systems/SPL assets. There could also be different standardized categories of assets (such as the environment, information systems, services, components and information or data) to make easier and more systematic the assets valuation. Dependencies between assets could also exist, so that valuations are propagated through the dependency tree of assets and therefore only the higher assets in the dependency tree have to be explicitly valued, the other assets would have the 'accumulated value' (which is defined as the highest value among it and any ones above). For example, assume that the *actor* "company ECMA" has a *goal* "provide payment by credit card service" which is an asset, and constrained by the *Security Constraint* "Keep data confidentiality", so that it is also related to the *softgoal* "confidentiality", and the *Security Constraint* is assigned an '*asset value*' of 7. This asset depends on the *secure resource* "web-server-SSL" (which is also an asset), due to the relations of the meta-model that implies the "web-server-SSL" '*asset value'* will be at least 7 (the accumulated value) according to the risk assessment methodology we followed (Magerit [12]).

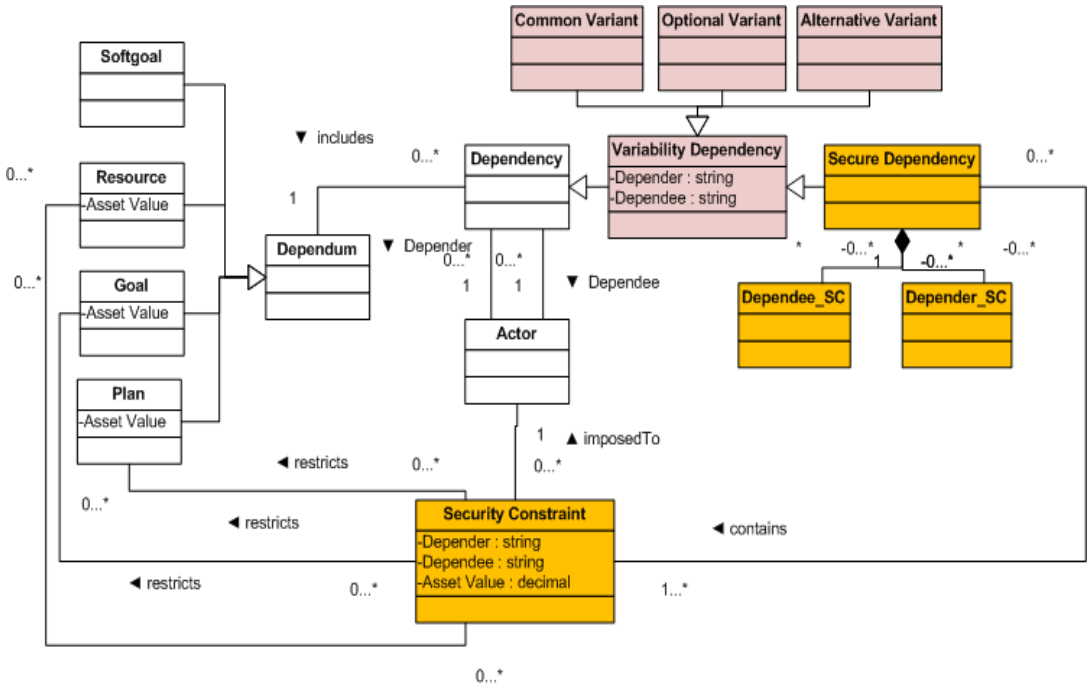

Fig. 2 Extension to SEAM ('SEAM-SPL')

[1] Asset: Anything that has value to the organization (ISO/IEC 13335)

In order to manage the variability of the SPL and the instantiated applications from the SPL at the level of a particular *Actor*, we extended the SEGM of Secure Tropos. The extension shown in Fig. 3 consists of adding an entity named '*Variation'* which could have as value: 'common', 'optional' or 'alternative', and which is related to the entities *Goal*, *Security Constraint*, *Plan* and *Resource* by means of a relationship '*is part of*' and which an *Actor* could have several *Variation*. It represents the variation object and defines a concrete type of variation (''how does it vary?").

The starting points of the variability modeling according to our proposed process in next section are the goals and next secure goals, because if the variability and traceability links are carefully established, they allow us to decide what security goals are needed to maintain the security aligned with the goals of the SPL or product/application and what the optimal set of security constrains of a determined priority according to the security risks 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 variations or variants selection process, and the selection is made in the requirements level rather than in the design level.

Finally, with the aim of providing a security risk criterion during the security requirements engineering in SPL engineering at the level of a particular *Actor*, we have also added a new entity: '*Threat*', which has as attributes (according to the Magerit [12]): '*Degradation'*, '*Likelihood'*, '*Impact'* and '*Risk'*. We use the definition of threat as a potential cause of an unwanted incident, which may result in harm to a system or organization [9]. Hence, the assets are exposed to threats which may prevent the security goals from being achieved. In a SPL, not all threats affect all assets nor all their security goals, so those which are common and optional have to be identified. To calculate the '*impact'* of each threat on the assets, the *asset values* of each security constraint along with the '*degradation'* caused by the threat on the assets (which must be estimated by the security risk expert within a range from 0 to 100%) are taken into account (Impact = round(accumulated value x degradation)). The *impact* and the '*likelihood'* of occurrence or rate of occurrence of the threat (which must be also estimated by the security risk expert) are taken into account in order to calculate the '*risk'* according to a defined formula in Magerit ($R(V_i, F_j) = V_{i+j-n}$)) ['R' is risk, 'V' is asset value, 'F' is likelihood]. The risk[2] is then classified in a range of 0 to 5 (according to the Magerit [12] scale). So that any estimation of impact and risk are "potential" if no '*secure plans*' are deployed.

[2] Risk is an estimate of the degree of exposure to threat to one or more assets causing damage or prejudice to the organization

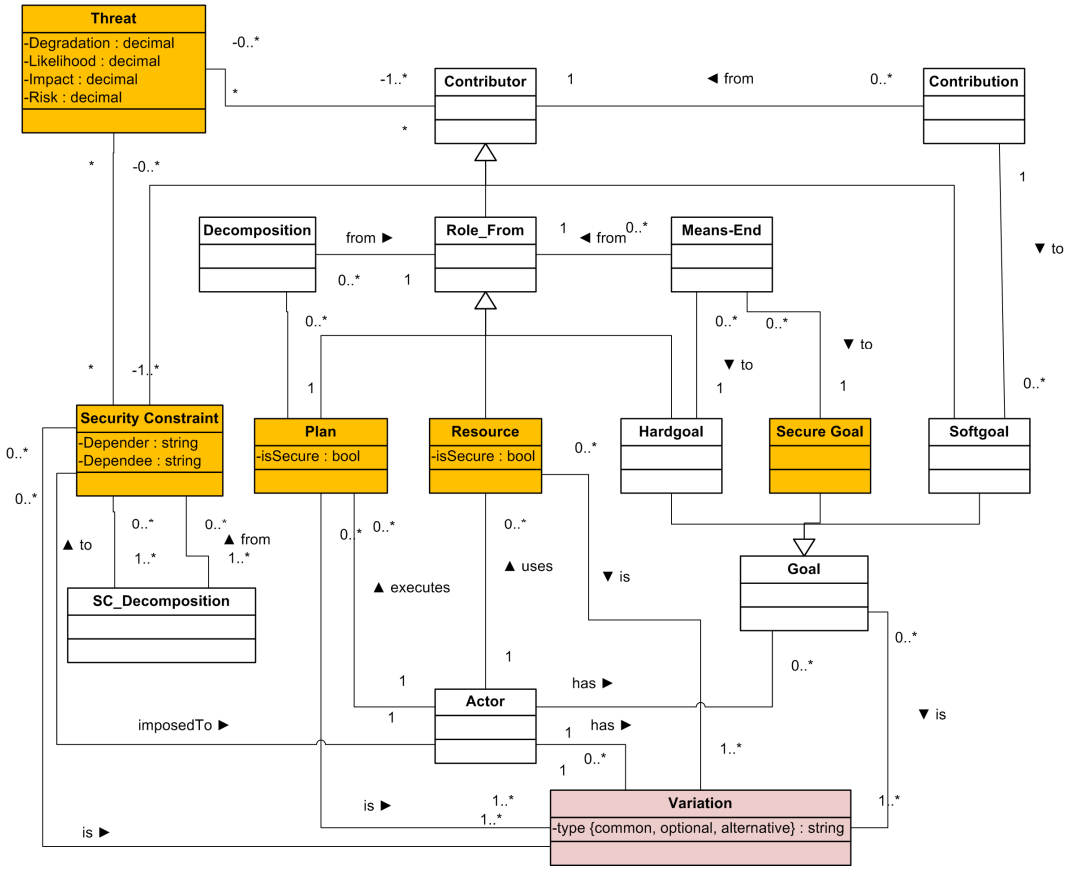

Fig. 3 Extension to SEGM ('SEGM-SPL')

4.4. Secure Tropos Process Extension

'Secure Tropos – SPL' process is an iterative and incremental process, which is an add-in of activities and 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. 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. 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.

We have specified these new tasks and activities of the process using the OMG standard SPEM 2.0 [20]. 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. 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 'Secure Tropos – SPL' process, to enhance its reuse and to facilitate its integration into other processes and frameworks. Thus, by using SPEM in the 'Secure Tropos – SPL' 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 'Secure Tropos – SPL' into the process map of their organization or company.

In accordance with SPEM, SREPPLine is described by using the structure shown in Fig. 4. 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.*

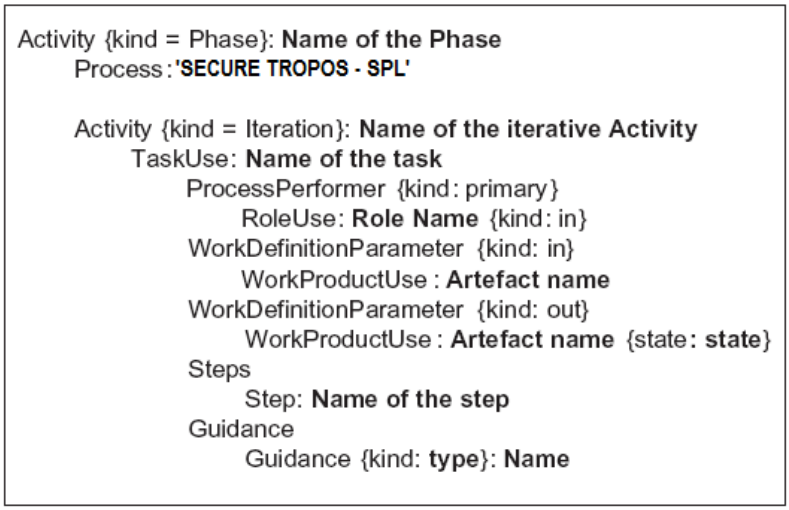

Fig. 4 'Secure Tropos - SPL' structure using SPEM 2.0

As shown in Fig. 1 and in Fig. 5, 'Secure Tropos – SPL' is composed of two activities: the Secure Tropos Domain Requirements Engineering (STDReq) activity (A1) and the Secure Tropos Application Requirements Engineering (STAReq) activity (A2).

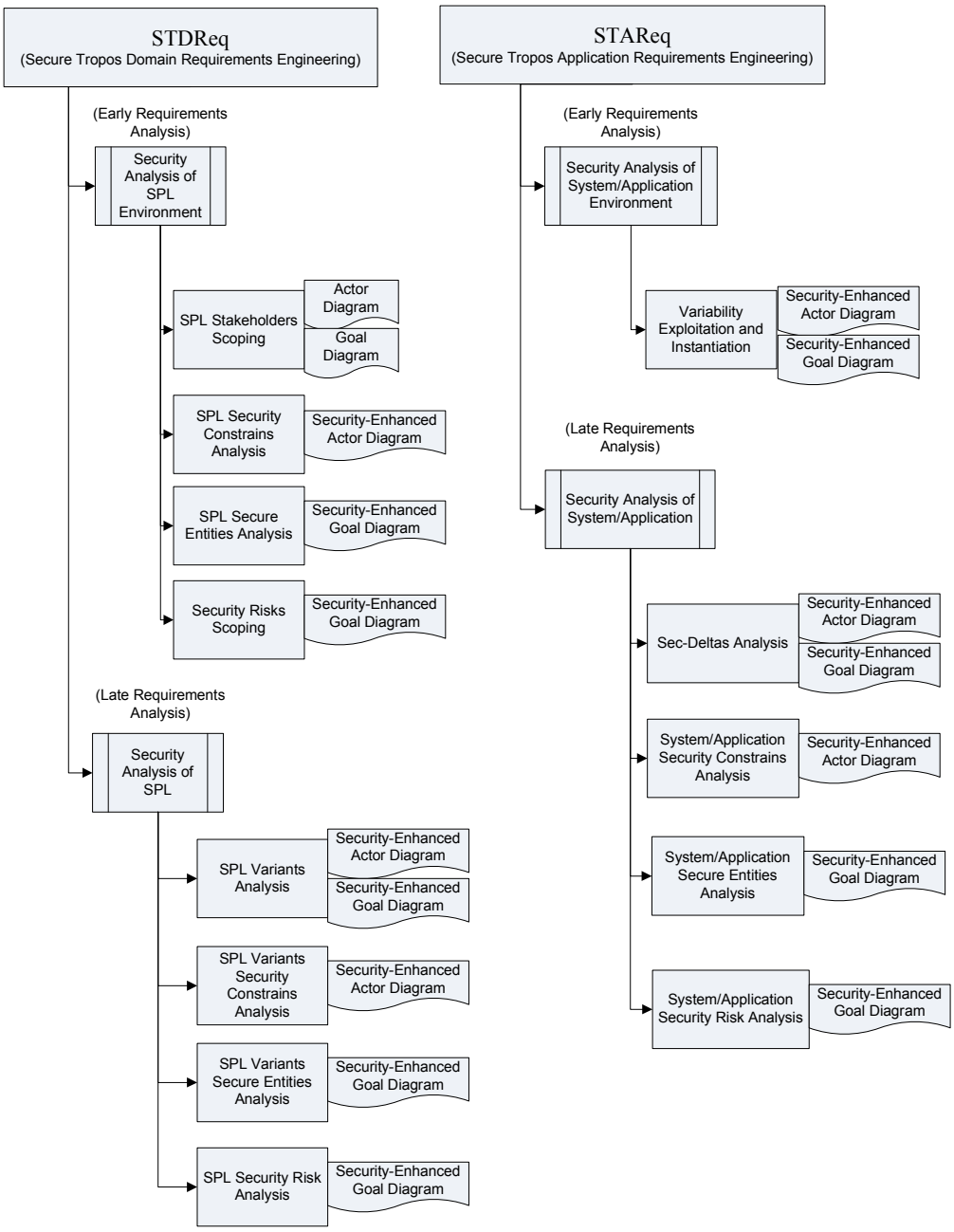

Fig. 5 'Secure Tropos – SPL' process overview

4.4.1. Secure Tropos Domain Requirements Engineering – (STDReq)

The main aim of this activity is the development of common and variable security requirements and related security artefacts of the SPL. The details of this activity are shown in Table 1.

Table 1 STDReq activity specified with SPEM (OMG)

Activity {kind = Phase}: Domain Requirements Engineering
Process: Secure Tropos – SPL
Activity {kind = Iteration}: **Secure Tropos Domain Requirements Engineering (STDReq) – (A1)**

TaskUse: A1.1 - **Security Analysis of Software Product Line Environment**

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: **Goal model (Tropos)**
WorkProductUse: **Organisation security policy**
WorkProductUse: **Law and regulations**
WorkProductUse: **Requests for additional / altered security constrains**

WorkDefinitionParameter {kind: out}
WorkProductUse: **List of common goals of the SPL** {state**: initial**}
WorkProductUse: **Security Enhanced Actor Diagram of the SPL – 'SEAM-SPL'** {state**: initial**}
WorkProductUse: **Security Enhanced Goal Diagram of the SPL – 'SEGM-SPL'** {state**: initial**}
WorkProductUse: **Table with the Security Risks of the SPL** {state**: initial**}

Steps
Step: A1.1.1 **SPL stakeholders scoping**
Step: A1.1.1.1 **Identify common actors of the SPL**
Step: A1.1.1.2 **Analyze the requests for additional / altered common goals**
Step: A1.1.2 **SPL security constrains analysis**
Step: A1.1.2.1 **Identify common goals**
Step: A1.1.2.2 **Identify security constrains**
Step: A1.1.2.3 **Security constraint modelling**
Step: A1.1.3 **SPL secure entities analysis**
Step: A1.1.3.1 **Identify security considerations imposed by the environment of the SPL**
Step: A1.1.3.2 **SPL secure entities and secure capability modelling**
Step: A1.1.4 **Security risk scoping**
Step: A1.1.4.1 **Common assets valuation**
Step: A1.1.4.2 **Common threats identification and calculation of degradation, likelihood and impact of each threat**
Step: A1.1.4.3 **Common risks assessment**
Step: A1.1.4.4 **Perform a security balance analysis of the SPL**
Step: A1.1.5 **Inspect 'SEAM-SPL' and 'SEGM-SPL' models**

Guidance
Guidance {kind: **Practice**}: **Questionnaire**
Guidance {kind: **Practice**}: **Interviews**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Application goals matrix**
Guidance {kind: **Practice**}**: Security constraint modelling (Secure Tropos)**
Guidance {kind: **Practice**}**: Secure entities modelling (Secure Tropos)**
Guidance {kind: **Practice**}**: Secure capability modelling (Secure Tropos)**
Guidance {kind: **Checklist**}: **Organization policy, laws and standards**

Activity {kind = Phase}: Domain Requirements Engineering
Process: Secure Tropos – SPL
Activity {kind = Iteration}: **Secure Tropos Domain Requirements Engineering (STDReq) – (A1)**

TaskUse: A1.2 - **Security Analysis of Software Product Line**

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 of each variant**
WorkProductUse: '**SEAM-SPL'**
WorkProductUse: '**SEGM-SPL'**
WorkProductUse: **Organisation security policy for each variant**
WorkProductUse: **Law and regulations for each variant**
WorkProductUse: **Requests for variant goals & security goals**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Security Enhanced Actor Diagram of the SPL – 'SEAM-SPL'** {state**: initial**}
WorkProductUse: **Security Enhanced Goal Diagram of the SPL – 'SEGM-SPL'** {state**: initial**}
WorkProductUse: **Table with the Security Risks of all the variants of the SPL** {state**: initial**}

Steps
Step: A1.1.1 **SPL variants analysis**
Step: A1.1.1.1 **Identify and model actors which are variants**
Step: A1.1.1.2 **Analyze the requests for variant goals & security goals**
Step: A1.1.2 **SPL variants security constrains analysis**
Step: A1.1.2.1 **Identify and model variant goals**
Step: A1.1.2.2 **Identify variant security constrains for each goal**
Step: A1.1.2.3 **Security constraint variability modelling**
Step: A1.1.3 **SPL variants secure entities analysis**
Step: A1.1.3.1 **Identify security considerations imposed by each variant of the SPL**
Step: A1.1.3.2 SPL **secure entities variability and secure capability modelling**
Step: A1.1.4 **SPL Security risk scoping**
Step: A1.1.4.1 **Variant assets valuation**
Step: A1.1.4.2 **Variant threats identification and calculation of degradation, likelihood and impact of each threat**
Step: A1.1.4.3 **SPL risks assessment (common & variant elements)**
Step: A1.1.4.4 **Perform a security balance analysis of all the variants of the SPL**
Step: A1.1.5 **Inspect 'SEAM-SPL' and 'SEGM-SPL' models**

Guidance
Guidance {kind: **Practice**}: **Questionnaire**
Guidance {kind: **Practice**}: **Interviews**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Application goals matrix**
Guidance {kind: **Practice**}: **Security constraint modelling** (Secure Tropos)
Guidance {kind: **Practice**}: **Secure entities modelling** (Secure Tropos)
Guidance {kind: **Practice**}: **Secure capability modelling** (Secure Tropos)
Guidance {kind: **Checklist**}: **Organization policy, laws and standards**

4.4.2. Secure Tropos Application Requirements Engineering – (STAReq)

The main aim of this activity is the elicitation and documentation of the security requirements and their related security artefacts in the SPL application and reusing the security domain artefacts and requirements as far as possible. The details of this activity are depicted in Table 2.

Table 2 STAReq activity specified with SPEM (OMG)

Activity {kind = Phase}: Domain Requirements Engineering
Process: Secure Tropos – SPL
Activity {kind = Iteration}: **Secure Tropos Domain Requirements Engineering (STDReq) – (A2)**

TaskUse: A2.1 - **Security Analysis of System / Application Environment**

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: **Security Enhanced Actor Diagram of the SPL – 'SEAM-SPL'**
WorkProductUse: **Security Enhanced Goal Diagram of the SPL – 'SEGM-SPL'**
WorkProductUse: **Application specific environment, policies and regulations**

WorkDefinitionParameter {kind: out}
WorkProductUse: **List of common goals of the SPL** {state**: initial**}
WorkProductUse: **Security Enhanced Actor Diagram of the Application – 'SEAM-SPL'** {state**: initial**}
WorkProductUse: **Security Enhanced Goal Diagram of the Application – 'SEGM-SPL'** {state**: initial**}
WorkProductUse: **Application's stakeholder goals that do not correspond to domain goals** {state**: initial**}

Steps
Step: A2.1.1 **Variability exploitation and instantation**
Step: A2.1.1.1 **Define security goals of the application**
Step: A2.1.1.2 **Communicate the relevant variants to the stakeholders of the application**
Step: A2.1.1.3 **Inherit common variants and analyze alternative and optional variants**
Step: A2.1.1.4 **Select the appropriate variants and model the chosen variants: security constraint modelling, secure entities modelling and secure capability modelling**
Step: A2.1.1.4 **Collect the application's stakeholder goals that do not correspond to domain goals**

Guidance
Guidance {kind: **Practice**}: **Interviews**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}**: Security constraint modelling (Secure Tropos)**
Guidance {kind: **Practice**}**: Secure entities modelling (Secure Tropos)**
Guidance {kind: **Practice**}**: Secure capability modelling (Secure Tropos)**
Guidance {kind: **Checklist**}: **Application specific policy, laws and standards**

Activity {kind = Phase}: Application Requirements Engineering
Process: Secure Tropos – SPL
Activity {kind = Iteration}: **Secure Tropos Application Requirements Engineering (STAReq) – (A2)**

TaskUse: A2.2 - **Security Analysis of System / Application**

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 of each variant**
WorkProductUse: '**SEAM-SPL' of the application**
WorkProductUse: '**SEGM-SPL' of the application**
WorkProductUse: **Organisation security policy for each variant**
WorkProductUse: **Law and regulations for each variant**
WorkProductUse: **Requests for variant goals & security goals**

WorkDefinitionParameter {kind: out}
WorkProductUse: **Security Enhanced Actor Diagram of the application – 'SEAM-SPL'** {state**: initial**}
WorkProductUse: **Security Enhanced Goal Diagram of the application – 'SEGM-SPL'** {state**: initial**}
WorkProductUse: **Table with the Security Risks of the application** {state**: initial**}

Steps
Step: A2.1.1 **Sec-deltas analysis**
Step: A1.1.1.1 **Identify sec-deltas**
Step: A2.1.2 **System/application security constrains analysis**
Step: A1.1.2.1 **Application security constraint variability modelling**
Step: A2.1.3 **System/application secure entities analysis**
Step: A1.1.3.1 **Identify security considerations imposed by the needs of the application**
Step: A1.1.3.2 SPL **Application secure entities variability and secure capability modelling**
Step: A2.1.4 **System/application security risk scoping**
Step: A1.1.4.1 **Application assets valuation**
Step: A1.1.4.2 **Application threats identification and calculation of degradation, likelihood and impact of each threat**
Step: A1.1.4.3 **Application risks assessment**
Step: A1.1.4.4 **Perform a security balance analysis of the application**
Step: A2.1.5 **Inspect 'SEAM-SPL' and 'SEGM-SPL' models of the application**

Guidance
Guidance {kind: **Practice**}: **Interviews**
Guidance {kind: **Practice**}: **Meetings**
Guidance {kind: **Practice**}: **Security constraint modelling** (Secure Tropos)
Guidance {kind: **Practice**}: **Secure entities modelling** (Secure Tropos)
Guidance {kind: **Practice**}: **Secure capability modelling** (Secure Tropos)
Guidance {kind: **Checklist**}: **Application specific policy, laws and standards**

5. Example of application

A simple and short example related to health and social care SPL is outlined in this section in order to describe and show throughout the example the applicability of our proposed extension of Secure Tropos for SPL engineering (named 'Secure Tropos – SPL').

Details of the organization in which the case study presented herein was carried out will not be provided for reasons of confidentiality and the potential threat to its security as well as security technical details related to the project. Moreover, all the information regarding the information systems mentioned in this case study have been previously published in various public forums.

We will apply our approach to specify the security requirements of a software product line of a CRM (Customer Relationship Management) system, which may have several different configurations for three different public institutions of the public social security system of Spain. Therefore, we will characterize the system, named eCRM, as a SPL whose members vary by system configuration yet retain the same core functionalities. Obviously, this case study has to be simplified and summed up to enable points of our approach to be easily illustrated in this paper.

Graphically, as shown in Fig. 6, Fig. 7 and Fig. 8, in the SEAD (Security Enhanced Actor Diagram) and SEGD (Security Enhanced Goal Diagram) the '*Variability Dependencies*' are represented with '◄*V*' over the dependency that joins the entities, so that the tip of the triangle indicates the "owner" of the variant, i.e. '*Depender*' or '*Dependee*'. In addition, if an entity is a '*Variation*', it is depicted with a '*(V)*' within the representation of the entity. The Secure Tropos entities are represented in the figures as follow: an actor with a circle; a goal with rounded rectangle; a security constraint with an octagon; a plan with a hexagon; a resource with a rectangle; and a threat with a pentagon.

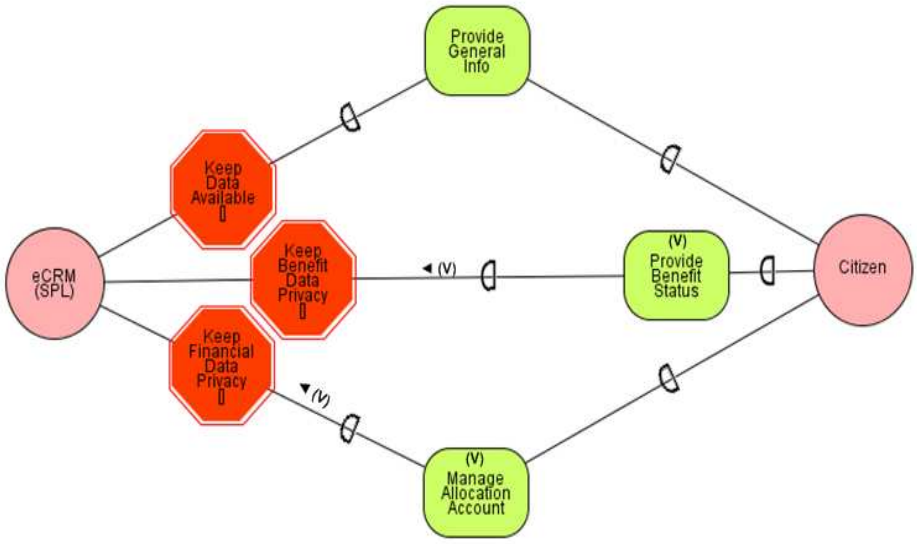

Fig. 6 Part of the SEAD of eCRM (SPL) – (Late Requirements phase)

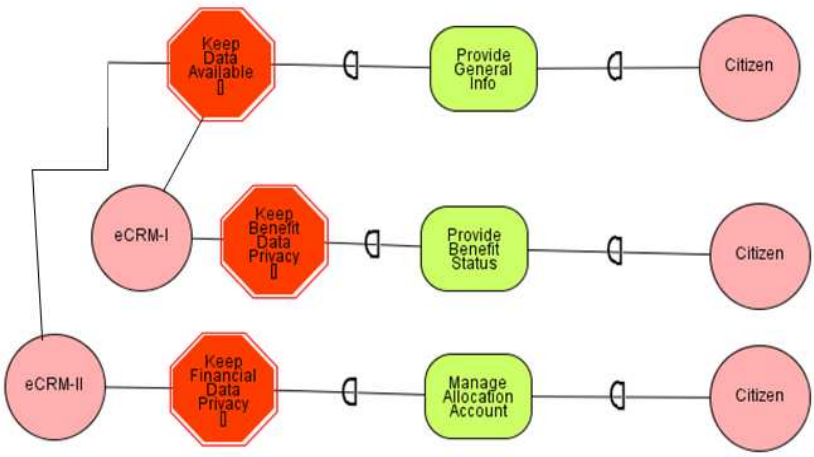

Fig. 7 Part of the SEAD of the instantiated applications of the eCRM (SPL) – (Late Requirements phase)

Fig. 6 shows a SEAD at the Late Requirements phase, which identifies and analyses the actors of the SPL and its environment. It also models the SPL's business goals, at business and service level, as well as it illustrates the analysis of the variability dependencies of these goals. This means that it supports the modeling of the variability of the goals, specifying the variant goals as common, optional or alternative. As shown in Fig. 7, the *actor* 'eCRM (SPL)' has strategic goals and intentions. In this example, the 'eCRM (SPL)' has a common service goal to citizens: "Provide general information about social security issues" and two optional service goals: "Provide the status of a citizen's benefit" and/or "Manage the allocation account contribution to the Social Security". In order to deal with the security issues, security constrains are introduced along with the variability dependencies. Security constrains, such as those shown in the model ("Keep data available", "Keep financial data privacy" and "Keep benefit data privacy"), represent restrictions related to security that the SPL must have and instantiated products must respect.

Fig. 7 illustrates a SEAD at the Late Requirements phase, which models the instantiation of applications from a SPL. In particular, the model represents two applications (eCRM-I and eCRM-II) instantiated from 'eCRM (SPL)', both of which inherit the common goals, constraints, plans and resources. Each application inherits the common business goals from the SPL and the stakeholders of each application choose the optional business goals by exploiting the variability of the eCRM(SPL).

Furthermore, the SEGD shown in Fig. 8 allows a deeper understanding of how the SPL reason about goals to be fulfilled, plans to be performed and availability of resources. It completes the SEAD with the reasoning that each actor makes about its internal goals and constraints, plans and resources. It can be seen that each variant business goal, which is restricted by a constraint, has related secure goals, which satisfy the constraint by means of secure plans that need resources.

Finally, through the entity '*Variation*' of the SEGM it is possible to trace the entities which are part of an instantiated application from the SPL. Hence, by means of this entity of the SEGM we can identify that for the instantiated application eCRM-I, the secure variant entities that are part of it are: the resource 'Benefit database'; the secure goals 'System privacy ensured' and 'User

authenticity ensured'; the secure plans related to these secure goals 'Crypto protocol' and 'User authentication'.

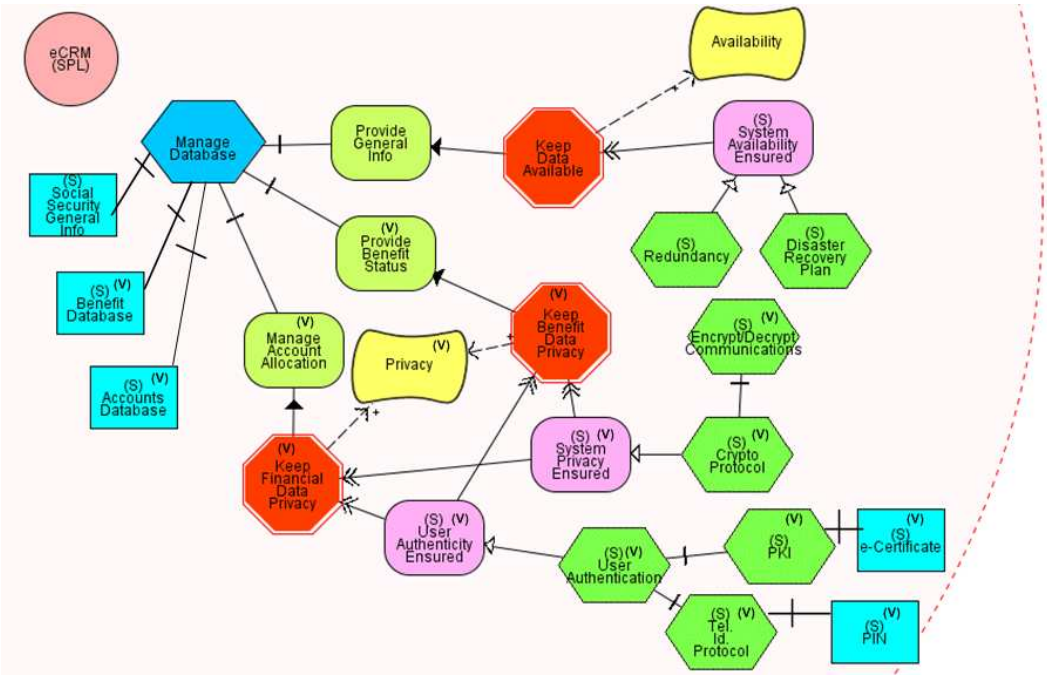

Fig. 8 Part of the SEGD of eCRM(SPL) - (Late Requirements phase)

6. Conclusions and Future Work

A large number of goal-oriented requirements engineering approaches have been proposed in the literature, which focus on eliciting security requirements. However, most of these approaches provide little help as how security requirements can be elicited and modelled in the context of SPL, at both the social and technical dimension, along with the fact that many standard requirements engineering practices must also be appropriately tailored to the specific demands of SPL [3].

This paper introduces the foundations of an approach that fills this gap by proposing *SecureTropos-SPL*, an extension to Secure Tropos to support SPL. The contribution of this work is that of: explaining how SPL concepts are aligned with Secure Tropos concepts; the presentation of a risk-driven goal-based process as a redefinition of the Secure Tropos process for SPL engineering; and an extension of Secure Tropos metamodel and language to support 'variability' modelling and 'risk' elements. Hence, by means of this approach it is possible to elicitate and to analyze both social and technical security requirements from the early stages of the SPL development process based on security goals and following a security risk criterion.

As future work, we plan to provide appropriate tool support to our approach. This will enable us to apply our work to large and complex case studies and explore its integration with relevant design-level proposals (such as UMLSec in [18]) to facilitate the secure design of SPL.

Acknowledgements

This research is part of the following projects: MEDUSAS (IDI-20090557) and ORIGIN (IDI-2010043(1-5), financed by the Centre for Industrial Technological Development (CDTI) and the FEDER, MAGO-PEGASO (TIN2009-13718-C02-01) awarded by the Spanish Ministry for Science and Technology and SERENIDAD (PEII11-0327-7035) and SISTEMAS (PII2I09-0150-3135) financed by the Council of Education and Science of the Castilla-La Mancha Regional Government.

References

1. J.L. Arciniegas, J.C. Dueñas, J.L. Ruiz, R. Cerón, J. Bermejo, and M.A. Oltra, *Architecture Reasoning for Supporting Product Line Evolution: An Example on Security*, in *Software Product Lines: Research Issues in Engineering and Management*, T. Käkölä and J.C. Dueñas, Editors. 2006, Springer.
2. J. Bayer, S. Gerard, O. Haugen, J. Mansell, B. Moller-Pedersen, J. Oldevik, P. Tessier, J.-P. Thibault, and T. Widen, *Consolidated Product Line Variability Modeling*, in *Software Product Lines: Research Issues in Engineering and Management*, T. Käkölä and J.C. Dueñas, Editors. 2005. p. 195-241.
3. A. Birk and G. Heller, *Challenges for requirements engineering and management in software product line development.* International Conference on Requirements Engineering (REFSQ 2007), 2007: p. 300-305.
4. J. Bosh, *Design & Use of Software Architectures*. 2000: Pearson Education Limited.
5. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, *Tropos: Agent-Oriented Software Development Methodology*. 2004: Journal of Autonomous Agents and Multi-Agent System. p. 203-236.
6. P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. 2002: Addison-Wesley.
7. B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, *A comparison of security requirements engineering methods.* Requirements Engineering, 2009. **15**: p. 7-40.
8. T.E. Faegri and S. Hallsteinsen, *A Software Product Line Reference Architecture for Security*, in *Software Product Lines: Research Issues in Engineering and Management*, T. Käkölä and J.C. Dueñas, Editors. 2006, Springer.
9. ISO/IEC, *ISO/IEC 13335 Information technology - Security techniques - Management of information and communications technology security*. 2004.
10. K. Kang, S. Cohen, J.A. Hess, W.E. Novak, and S.A. Peterson, *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. 1990, Software Engineering Institute, Carnegie-Mellon University.
11. J. Kim, M. Kim, and S. Park, *Goal and scenario bases domain requirements analysis environment*, in *The Journal of Systems and Software*. 2005. p. 926 - 938.
12. M.A.P., *Methodology for Information Systems Risk Analysis and Management (MAGERIT version 2)*. 2005, (Ministry for Public Administration of Spain).
13. J.D. McGregor, *Testing a Software Product Line*, in *Testing Techniques in Software Engineering*, P. Borba, et al., Editors. 2010, Springer. p. 104-140.
14. D. Mellado, C. Blanco, L.E. Sanchez, and E. Fernández-Medina, *A Systematic Review of Security Requirements Engineering.* Computers Standards & Interfaces 2010. **32**: p. 153-165.
15. D. Mellado, E. Fernández-Medina, and M. Piattini, *Security requirements engineering framework for software product lines.* Information and Software Technology, 2010. **52**: p. 1094-1117.
16. D. Mellado and H. Mouratidis, *Towards the Extension of Secure Tropos Language to Support Software Product Lines Development*, in *International Workshop on Security In Information Systems (WOSIS-2012)*. 2012. p. (accepted).
17. H. Mouratidis, *Secure Tropos: An Agent Oriented Software Engineering Methodology for the Development of Health and Social Care Information Systems.* International Journal of Computer Science and Security, 2009. **3**(3): p. 241-271.
18. H. Mouratidis and J. Jürjens, *From goal-driven security requirements engineering to secure design.* International Journal of Intelligent Systems, 2010. **25**(8): p. 813-840
19. OECD, *The promotion of a culture of security for information systems and networks in OECD countries*, in *DSTI/ICCP/REG(2005)1/FINAL*. 2005, Organisation for Economic Co-operation and Development.
20. OMG, *Software & Systems Process Engineering Meta-Model Specification v.2.0*. 2008: http://www.omg.org/spec/SPEM.
21. K. Pohl, G. Böckle, and F.v.d. Linden, *Software Product Line Engineering. Foundations, Principles and Techniques*. 2005, Berlin Heidelberg: Springer.
22. K. Schmid and I. John, *A Customizable Approach To Full-Life Cycle Variability Management*, in *Science of Computer Programming, Elsevier, 53*. 2004. p. 259-284.
23. K. Schmid, K. Krennrich, and M. Eisenbarth, *Requirements Management for Product Lines: A Prototype*. 2005, Fraunhofer IESE.
24. M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. *COVAMOF: A Framework for Modeling Variability in Software Product Families*. in *Proc. of the Third Softw. Product Line Conf. (SPLC 2004)*. 2004. Boston, MA, USA.