Managing Changes with Legacy Security Engineering Processes

Edith Felix, Olivier Delande
Thales
Palaiseau, France
{edith.felix,olivier.delande}@thalesgroup.com

Fabio Massacci, Federica Paci
Department of Information Engineering and Computer Science
University of Trento
Povo, Trento
{Fabio.Massacci,Federica.Paci}@unitn.it

Abstract— **Managing changes in Security Engineering is a difficult task: the analyst must keep the consistency between security knowledge such as assets, attacks and treatments to stakeholders' goals and security requirements. Research-wise the usual solution is an integrated methodology in which risk, security requirements and architectural solutions are addressed within the same tooling environment and changes can be easily propagated.**

This solution cannot work in practice as the steps of security engineering process requires to use artefacts (documents, models, data bases) and manipulate tools that are disjoint and cannot be fully integrated for a variety of reasons (separate engineering domains, outsourcing, confidentiality, etc.). We call such processes *legacy security engineering processes.*

In this paper, we propose a change management framework for legacy security engineering processes. The key idea is to separate concerns between the requirements, risk and architectural domains while keeping an orchestrated view (as opposed to an integrated view). We identify some mapping concepts among the domains so that little knowledge is required from the requirement manager about the other domains, and similarly for security risk manager and the system designer: they can stick to their well known (and possibly certified) internal process. This minimal set of concepts is the *interface* between the legacy processes. The processes are then orchestrated in the sense that when a change affects a concept of the interface, the change is propagated to the other domain.

We illustrate this example by using the risk modeling language (Security DSML) from Thales Research and the security requirement language (SI*) from the Univ. of Trento.

System and software engineering life cycle, Security engineering, Security risks, Requirements, Tooling

INTRODUCTION

Change management in security engineering is a particularly daunting task not only because of the inherently difficulty of the task but also for two concerning factors that characterize modern production in an industrial environment.

System and software engineering in industry is a complex process that is subject to many standard and certification processes, in particular when the critical security infrastructures is at stake.

The need to show compliance with standards e.g ISO 15288 and ISO 12207, respectively for system and software engineering makes often the engineering process quite rigid.

Such rigidity is further increased when those security aspects must be further taken into account. Security standards or best practices must be considered such as ISO 27000, EBIOS,CORAS, CRAMM, OCTAVE, BSIMM [16-20]. The design process must also be compliant with those standards.

For complex systems the security engineering process is also inevitably supported by artefacts (UML models of the system to be, DOORS format for requirements [9], UML risk profiles in CORAS [17] etc), and large companies tend to adapt and customize these artefacts to fit their needs and application domains [14,15]. The combination of these two factors makes each step of security engineering process highly customized and highly rigid and de facto unchangeable, as the switching cost would be too high. We end up with the combination of *legacy software engineering processes.*

So what happens when a security requirement or a threat model changes? For example, in the air traffic management domain, 9/11 has dramatically changed the threat model and implied a different design of the "interface" between cabin and cockpit. Changes must percolate through these structures and they might not get through completely. The solutions proposed by most researchers is to have a unique process integrating security requirements, risk assessment and security architectures [1,2,3].

Contribution of the paper

In this paper we propose a security engineering process where the presence of proprietary steps is not a liability. We focus our attention on the interactions between the *security risk manager*, the *requirement manager*, and the *system designer* and we show how the activities performed by these stakeholders can be orchestrated. The key feature of the orchestrated process is *separation of concern principle*. An important advantage of separation of concern is that in-depth expertise in the respective domains is not a prerequisite. The

978-1-61284-4577-0085-9/11/$26.00 ©2011 IEEE

orchestrated process allows the separate domains to leverage on each other without the need of full integration. As a counterpart, consistency of concerns should be ensured. We assume that security risk manager, the requirement manager, and the system designer share a minimal set of concepts which is the interface between their respective processes: each process is conducted separately and only when a change affects a concept of the interface, the change is propagated to the other domain.

The paper is structured as follows. Section II introduces the running example based on the evolution of ATM systems that is taking place as planned by the Single European Sky ATM Research (SESAR) Initiative. In Section III we instantiate the requirement and the security and system domains with SI* and Security DSML modeling languages respectively. In Section IV we present the interface between the system engineering process and the risk analysis process. In Section V, we outline the importance of including risk analysis into system engineering process and we illustrate a security engineering process based on the collaboration between the security risk manager, the requirement manager, and the system designer. In section VI, we illustrate the orchestrated process based on the running example in Section II. Section VII presents related works. Section VIII concludes the paper.

II. RUNNING EXAMPLE

To illustrate the change propagation process, we will focus on the ongoing evolution of Air Traffic Management (ATM) systems planned by the ATM 2000+ Strategic Agenda and the Single European Sky ATM Research (SESAR) Initiative [4].

Part of ATM system's evolution process is the introduction of a new decision support tool for air traffic controllers (ATCOs) called Arrival Manager (AMAN) in order to support higher traffic loads. The main goal of the AMAN is to help ATCOs to manage and better organize the air traffic flow in the approach phase. The introduction of the AMAN requires new operational procedures and functions and imposes new security properties to be satisfied. Before the addition of the AMAN, the Sector Team[1] had to manually perform the operations related to the approach phase: the generation of the arrival sequence and the allocation of runaways. Now, some of the operations that were manually done by Sector Teams are performed by the AMAN such as providing sequencing and metering capabilities for runways, airports or constraint points, creating an arrival sequence using 'ad hoc' criteria, managing and modifying proposed sequences, supporting runway allocation at airports with multiple runway configurations, and generating advisories for example on the time to lose or gain, or on the aircraft speed. The introduction of the AMAN requires also the addition of a new role between ATCOs, called Sequence Manager (SQM), who will monitor and modify the sequences generated by the AMAN and will provide information and updates to the Sector Team.

[1] The sector team consists of a Tactical Controller and a Planning Controller.

III. BACKGROUND

We instantiate the requirement framework to SI* [3] and the risk framework to Security DSML [13].

SI* is a requirement framework which supports both early and late requirement analysis. SI* has several extensions, but in this paper we focus on the trust and risk extension proposed in [2]. We only consider a subset of SI* relations, namely *AND/OR decomposition*, *means-end*, *require*, *request*, and *dependency* and *trust relations*. We also consider the *business object* [5] concept which is a combination of goals, processes, and resources.

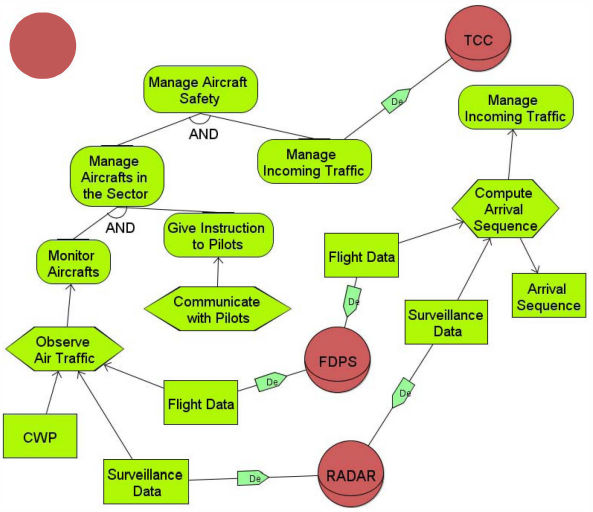

Figure 1. Example of SI* model

The requirement analysis consists of five steps: 1)Identify relevant stakeholders, modeled as *actor* (circle) and its structure; 2) Capture and refine actor's *goals* (rounded rectangle); 3) Define means - i.e., *process* (hexagon) or *resource* (rectangle) - to achieve their goals; 4) Model strategic dependencies between actors in fulfilling/executing/providing some goals/processes/resources; 5) Model specific aspects such as security or risk:
e.g introduce *security goals*, which are goals concerning the fulfillment of security properties [1] or assess the achievement level of high-level goals, such as risk level [2].

The requirement analysis is an iterative process that aims at refining the stakeholders' goals until all goals are achieved.
The results of the analysis process are captured by a SI* model as the one in Figure 1 illustrating the running example introduced in Section 2. The model consists of four actors: Planning Controller (PLC), Tactical Controller (TCC), Radar and Flight Data Processor (FDPS). The PLC has one main goal that is Manage Aircraft Safety that is decomposed into Manage Aircraft in the Sector and Manage Incoming Traffic subgoals. The latter goal is delegated to TCC who fulfills by providing the process

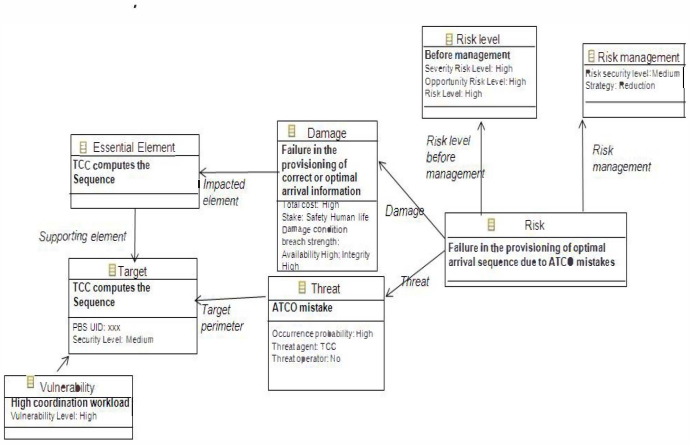

Figure 2. An example of Security DSML model

Compute Arrival Sequence. This task requires the resources Flight Data and Surveillance Data that are provided by the FDPS and the Radar respectively.

Security DSML is the language and a tool developed to capture the security risk analysis concepts derived from the French EBIOS methodology [16]. As a tool, Security DSML realizes a Viewpoint of a system Architecture Model as defined in coming ISO 42010 standard [23].

The main security concepts are the following:

- *Essential element*: an element of the system at Business Architecture or Service-oriented Architecture Plans.
- *Damage*: the impact related to a risk on the essential elements of system.
- *Target*: an element of the system potentially threatened by one or more threats.
- *Vulnerability*: weakness in a system, system security procedures, internal controls, or implementation that could be exploited.
- *Threat*: any circumstance or event with the potential to adversely impact a system through unauthorized access, destruction, disclosure, modification of data, and/or denial of service.
- *Risk*: possibility that a particular threat will adversely damage an element of the system design.
- *Security objective*: expression of the intention to counter identified risks by goals regarding the security of the system.
- *Security requirement*: a functional or assurance general specification covering one or more security objectives.
- *Security solution*: a security measure that implements a security requirement.

Figure 2 shows the ATM example. The model starts from the activity TCC computes the sequence called an *Essential Element* in Security DSML language. A potential *Damage* Failure in the provisioning of correct or optimal arrival information is identified. The ATCO[2] as supporting elements of the activity, called *Targets* in Security DSML, are vulnerable to High coordination workload, and are subject to the *Threat* ATCO mistake. Then the *Risk* Failure in the provisioning of correct or optimal arrival information is identified, which has a high risk level, which needs to be reduced to at least medium.

IV. CONCEPUAL MAPPING

Even though conducted separately, the requirement analysis, and the risk analysis processes can be orchestrated so that they can benefit from the respective results. In order to allow the orchestration between these processes, we need to identify a set of concepts that is the *interface* between them (see Table I).

TABLE I. INTERFACE

	Conceptual Mapping		
Requirement	***Risk***	***Architecture***	***Type***
Business Object	Essential Element		Shared
Goal	Security Objective		Mapped
Security Goal	Security Requirement		Mapped
Process		Security Solution	Mapped

We distinguish the interface concepts in *shared elements* and *mappable elements*. The shared elements are model elements that conceptually have the same semantic in the three domains. The mappable elements are elements from one domain that are not shared by the other, but nevertheless can be mapped to elements of the other domain.

When a change affects a mappable or shared element in one domain such change is propagated to the other domain. The following table summarizes the conceptual mapping.

V. CHANGE MANAGEMENT PROCESS

International standards like ISO/IEC 15288:2008 and ISO/IEC 12207:2008 [21, 22] describe the system and software life cycle of the engineering process and including clauses mentioning that non-functional properties such as security should be considered in different phases.

In security specialty engineering, risk analysis methodologies such as EBIOS, CORAS or CRAMM serve security risk managers to produce a rationale for security requirements and assess the risks in an exhaustive way, as needed in domains such as administration or military systems. The risk management process does not cover the entire security engineering activities but is a key starting point to them.

Thus a first issue is to show how the risk management process and security requirement analysis can collaborate with the global system engineering process described in those

[2] ATCO stands for Air Traffic Controller, which here means Planning Controller (PLC) and Tactical Controller (TCC).

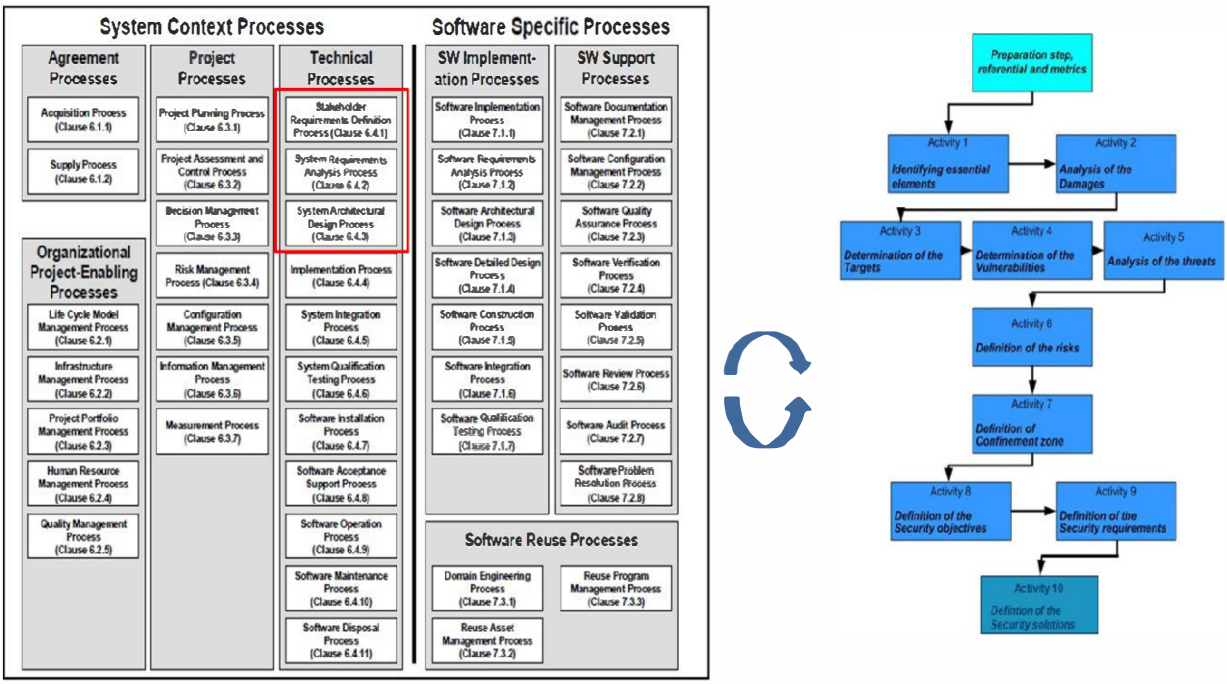

Figura 3. ISO/IEC 12207 vs EBIOS process

engineering standards. The difficulty resides in the necessary iterations needed to refine the security requirements since some vulnerabilities and risks will appear only once the system architectural design has been set up.

This article focuses on how the risk engineering process which has been standardized independently can be orchestrated into the overall system engineering process. In particular we investigate how the processes Stakeholder Requirements Definition (Clause 6.4.1), Requirements Analysis (Clause 6.4.2), and Architectural Design (Clause 6.4.3) processes of ISO/IEC 12207 can be orchestrated with EBIOS risk methodology activities (see Figure 3).

The resulting orchestrated process is represented in Figure 4. The stakeholder relationship[3] technical manager gets the needs and requirements from the stakeholders (Clause 6.4.1). He pushes the information related to security needs to the security risk manager who expresses the unwanted damages and defines the first security objectives. The stakeholder relationship technical manager validates the security objectives with the stakeholders and consolidates them before sending them to the requirement manager. Then the requirement manager consolidates them with requirements from other stakeholders and sends all the requirements to the system designer.

The system designer analyzes the requirements (Clause 6.4.2) and defines the functions of the system. Once this is done, the security risk manager updates the essential elements (Activity 1) based on the functions of the system, updates the damages (Activity 2), adds some security objectives (Activity 8), defines first security requirements (Activity 9) and sends them to the system designer and to the requirement manager.

[3] The stakeholder relationship manager in the majority of requirement engineering framework is called requirement manager

The system designer validates the requirements analysis with the design authority who propagates it.

The system designer proceeds to architectural design of the system, allocating functions to elements of the system (Clause 6.4.3). This new organization of the model of the system is analyzed by the security risk manager who evaluates carefully the risks and defines security solutions (Activities 3 to 10), and sends the updates of the security solutions to the system designer who consolidates the system design and validates the architectural design with the design authority who propagates it. Architectural design and updated requirements are propagated to the system engineering manager and the security engineering manager for them to complete the design at physical layer and implement it (Clause 6.4.3 and Clause 6.4.4).Once he has chosen the security solutions, the security engineering manager sends the information to the security risk manager for targets and vulnerabilities determination (Activities 3 and 4) and a full update of the risk management cycle (Activities 5 to 10).

The updates are passed through the security engineering manager to the system engineering manager. The system engineering manager validates the architectural design and the existing elements of the implementation with the design authority.

VI. APPLICATION TO THE ATM DOMAIN

Here we illustrate some of the steps of the integrated process that involves the security risk manager, the requirement engineer and the system designer, by using the evolution scenario introduced in Section II.

1) The stakeholder relationship technical manager and the security risk manager interact to identify an initial set of security objectives to be passed to the requirement manager.

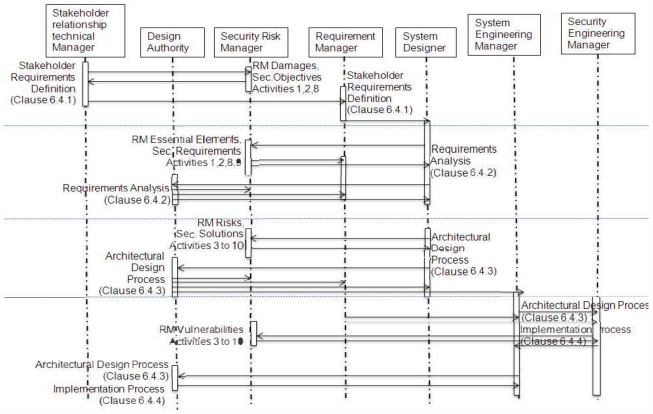

Figure 4.ISO/IEC 12207 sub-set of processes and EBIOS collaboration diagram

2) The stakeholder relationship technical manager passes the requirements proposed by the stakeholders and the initial security objectives to the requirement manager. A change request is triggered for the requirement domain: the SI* model illustrated in Figure 1 is produced by the requirement manager.

3) The system designer analyzes the SI* model provided by the requirement manager and then passes it to the security risk manager.

4) The security risk manager identifies the following new security objectives:
   - **O1** The system shall be computed automatically by an Arrival Manager system that covers the risk
     - **R1** Failure in the provisioning of correct or optimal arrival information due to ATCO mistakes.
   - **O2** The update of the system should be handled through a dedicated role of Sequence Manager that covers the risk **R1**.

   The above security objectives are refined into the following security requirements:
   - **RE1** The system should integrate an AMAN (refines security objective **O1**)
   - **RE2** The organization should integrate a SQM (refines security objective **O2**).

   Figure 5 represents the Security DSML model updated with the new security objectives and security requirements.

5) The changes into the Security DSML model trigger a change request for the requirement domain. The requirement manager receives the new security objectives and requirements and updates the SI* model as shown in Figure 6: two new actors, AMAN and the SQM have been added with their goals, process and resources.

6) The new processes Compute Arrival Sequence provided by AMAN and Monitor and Modify provided by SQM identified by the requirement manager has to be propagated to the system designer and to the security risk manager.

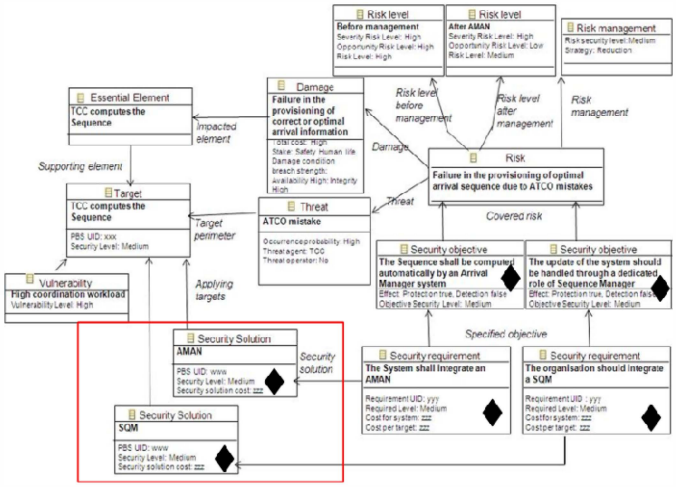

Figure 5. Security DSML Model after the introduction of AMAN

The security risk manager assesses the new processes proposed by the requirement manager and defines new security solutions to match the processes (outlined in red in Figure 5). Then, the security risk manager passes the identified security solutions to the system designer for validation.

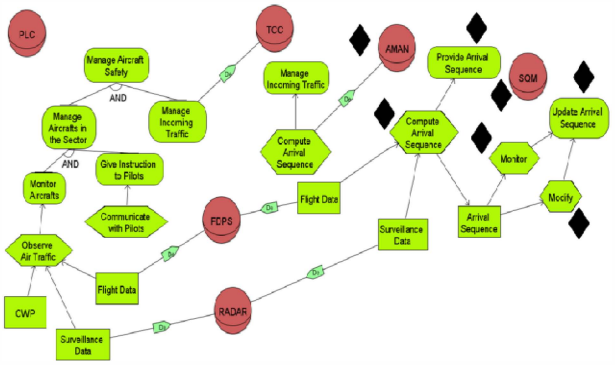

Figure 6. SI* Model after the introduction of the AMAN and SQM

VII. RELATED WORK

The predominant standards for system and software engineering are ISO/IEC 15288 and 12207 [21, 22]. Upcoming ISO/IEC 42010 [23] standard describes the common vocabulary and framework for working on several concerns and specialty engineering viewpoints which all refer to a common architecture description.

Among the security risk analysis methods CORAS [17] is based on UML environment and has proposed new techniques for structured diagrams. EBIOS [16] released a simplified EBIOS 2010 methodology which is more suitable for architectural engineering environment.

Existing requirement engineering proposals have been extended to include security concepts in the requirement

conceptual models and to support security related analysis. Van Lamswerde extended KAOS [1] by introducing the notion of obstacles to capture exceptional behaviours and anti-goals to model the intention of an attacker to threaten security goals. Massacci et al.[3] have defined Secure Tropos for modeling and analyzing authorization, trust and privacy concerns. Haley et al. [6] extend problem frames to determine security requirements for the system by considering possible security threats. Elahi et al. [7] extend i* with security related notions (e.g., attacker, vulnerability, malicious goal) for capturing and analyzing the impact of vulnerabilities to software systems. Asnar et al. [2] extend also i* with the notion of uncertain events and treatments to support the risk assessment process into requirement engineering process.

With respect to these proposals, our work does not require to extend existing requirement frameworks at modeling and process level but it just requires the requirement analyst and the risk analyst to share the understanding of a set of concepts to be able to communicate and share the results of the respective analysis processes. Moreover, our orchestrated process has also another advantage: it supports change propagation between the requirement and risk domain which is enabled by the shared interface.

In fact, only some requirement engineering proposals provide support for handling change propagation and for change impact analysis.

Chechik et al. [11] propose a model-based approach to propagate changes between requirements and design models that utilize the relationship between the models to automatically propagate changes. Hassine et al. [12] present an approach to change impact analysis that applies both slicing and dependency analysis at the Use Case Map specification level to identify the potential impact of requirement changes on the overall system. Lin et al. [10] propose capturing requirement changes as a series of atomic changes in specifications and using algorithms to relate changes in requirements to corresponding changes in specifications.

VIII. CONCLUSIONS

We have proposed a change management framework for legacy security engineering processes. The key idea is to separate concerns between the requirements, risk and architectural domains while keeping an orchestrated view. The orchestration has been based on the mapping of concepts among the domains so that little knowledge is required from the requirement manager about the other domains, and similarly for security risk manager and the system designer. The processes are then orchestrated in the sense that when a change affects a concept of the interface, the change is propagated to the other domain.

We have illustrated the framework using an example of evolution taken from the air traffic management domain. We are planning to apply the framework to other industrial case studies e.g the evolution of multi-application smart cards.

ACKNOWLEDGMENT

This work has been partly funded by EU project - Network of Excellence on Engineering Secure Future Internet Software (NESSoS) and by the EU-FP7-FET-IP-SecureChange project.

REFERENCES

1. A. van Lamsweerde, “Elaborating security requirements by construction of intentional anti-models,” in Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on, 2004, pp. 148–157.
2. Y. Asnar, P. Giorgini, and J. Mylopoulos, “Goal-driven risk assessment in requirements engineering,” Requirements Engineering, pp. 1–16, (to appear).
3. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, “Requirements engineering for trust management: model, methodology, and reasoning,” International Journal of Information Security, vol. 5, no. 4, pp. 257–274, Oct. 2006.
4. EUROCONTROL, “ATM Strategy for the Years 2000+,” 2003.
5. Y. Asnar, P. Giorgini, P. Ciancarini, R. Moretti, M. Sebastianis, and N. Zannone, “Evaluation of business solutions in manufacturing enterprises,” International Journal on Business Intelligence and Data Mining, vol. 3, no. 3, pp. 305 – 329, 2008.
6. C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, “Security requirements engineering: A framework for representation and analysis,” IEEE Trans. Softw. Eng., vol. 34, pp. 133–153, January 2008.
7. G. Elahi, E. Yu, and N. Zannone, “A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities,” Requirements Engineering, vol. 15, pp. 41–62, 2010.
8. L. Liu, E. Yu, and J. Mylopoulos, “Security and privacy requirements analysis within a social setting,” in Proceedings of the 11th IEEE International Conference on Requirements Engineering. Washington, DC, USA: IEEE Computer Society, 2003, pp. 151–161.
9. DOORS. http://www-01.ibm.com/software/awdtools/doors/.
10. L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett, “Introducing abuse frames for analysing security requirements,” in Proceedings of the 11th IEEE International Conference on Requirements Engineering. Washington, DC, USA: IEEE Computer Society, 2003, pp. 371–372.
11. M. Chechik, W. Lai, S. Nejati, J. Cabot, Z. Diskin, S. Easterbrook, M. Sabetzadeh, and R. Salay, “Relationship-based change propagation: A case study,” in Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering, ser. MISE ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 7–12.
12. J. Hassine, J. Rilling, and J. Hewitt, “Change impact analysis for requirement evolution using use case maps,” in Proceedings of the Eighth International Workshop on Principles of Software Evolution. Washington, DC, USA: IEEE Computer Society, 2005, pp. 81–90.
13. V. Normand, E. Félix, “Toward model-based security engineering: developing a security analysis DSML”, ECMDA-FA, 2009.
14. Eclipse Modeling Framework (EMF)http://www.eclipse.org/modeling/emf/.
15. Eclipse Graphical Modeling Framework (GMF)http://en.wikipedia.org/wiki/Graphical_Modeling_Framework.
16. EBIOS. http://www.ssi.gouv.fr/en/confidence/ebiospresentation.html
17. CORAS, http://coras.sourceforge.net/, SINTEF.
18. CRAMM, http://www.cramm.com, Siemens.
19. OCTAVE, http://www.cert.org/octave/, Carnegie Mellon.
20. BSIMM, Building Security In Maturity Model, http://bsimm.com/.
21. ISO/IEC 15288, Systems and software engineering — System life cycle processes, ISO, 2008.
22. ISO/IEC 12207, Systems and software engineering — Software life cycle processes, ISO, 2008.
23. ISO/IEC FCD 42010, Architecture description, draft.