Improved Security Requirements Engineering using Knowledge Representation

Yves Roudier (yves.roudier@eurecom.fr)*
Muhammad Sabir Idrees (Sabir.Idrees@eurecom.fr)*
Ludovic Apvrille (ludovic.apvrille@telecom-paristech.fr) †

Abstract:
We introduce in this paper a security meta-model for our SysML-Sec framework, developed to improve the security requirements engineering process through the explicit representation of security concerns with knowledge representation techniques. This meta-model enables the specification of ontological concepts which define the semantics of the security artifacts introduced through SysML-Sec diagrams. This meta-model also enables representing the relationships that tie several such concepts together. This representation is then used for reasoning about the knowledge introduced by system designers as well as security experts through the graphical environment of the SysML-Sec framework. In addition to its documentary aspect, such a meta-model makes it possible to introduce different types of verifications of security requirements and threats, and especially consistency checks regarding the content of all diagrams. We finally present a prototype that integrates meta-model descriptions into the SysML-Sec framework and its implementation using Semantic Web technologies.

Keywords: SysML, security, model driven engineering, knowledge representation.

1 Introduction

Most contributions around Model Driven Engineering (MDE) now offer appropriate methodologies and modeling environments for designing safe, complex, distributed, and real-time embedded systems. Yet security has long been considered only in retrospect, especially after serious flaws are discovered. We designed the SysML-Sec framework [AR13], an environment for developing embedded systems or distributed IT systems with embedded system components with an explicit focus on their lifecycle long engineering. Like most security requirements engineering approaches, the requirement process part in SysML-Sec considers (i) the identification of threats that may harm assets, and (ii) the elicitation of requirements originating from the application or mitigating those threats. Still, those two stages are subject to conceptual and terminological confusions, which may lead to the design of inadequate security mechanisms. Also, the consistency between multiple views of the same system is generally addressed manually in requirements engineering processes.

To address those problems, we propose to introduce knowledge representation and management techniques into the requirements engineering methodology. Those techniques

* EURECOM, Campus SophiaTech, CS 50193, 06904 Sophia Antipolis cedex France, Tel: 04.93.00.81.18, Fax: 04.93.00.82.00
† Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI, Campus SophiaTech, CS 50193, 06904 Sophia Antipolis cedex France, Tel: 04.93.00.84.06, Fax: 04.93.00.82.00

combine with SysML-Sec [AR13] models so that they can be integrated into the requirements engineering process.

2 A Knowledge Based Methodological Approach to Security Requirements Engineering

This section describes our unified methodology for the use of ontological concepts in a security requirement engineering process. Our aim is to devise an approach that extends the tool based process of engineering by guiding each activity within the process with the knowledge acquired in other activities.

2.1 Meta-modeling and approaches to security requirements engineering

There has been a quite remarkable progress in the area of security requirements engineering in the last 15 years. Surveys [NNY10, FGH+10, MBSFM10] discuss the many approaches proposed. Nhlabatsi et al. for instance classify them according to four dimensions, namely: (1) *goal-based approaches*, (2) *model-based approaches*, (3) *problem-oriented approaches*, and (4) *process-oriented approaches*. We contend that while this classification is useful, it mainly represents different perspectives over the same process of security requirements engineering. While adopting a single perspective is simpler for the practitioner, there are benefits in combining several approaches.

However, most if not all of these approaches are bound to specific modeling languages and tools that support them. Those languages and tools constitute at the same time the major obstacle to achieving such an endeavor. The use of meta-modeling can help increase the expressivity of security requirements engineering while retaining existing languages and tools as the primary medium for requirements input and visualization. The availability of a meta-model also makes it possible to automate verifications through the introduction of meta-level reasoning capabilities. We explain in the rest of this paper how we introduced a knowledge-centric meta-model, supported through the definition of security ontologies, which we integrated into our SysML-Sec platform. The next section describes in more detail the objectives of this meta-model.

2.2 Objectives of meta-modeling

We consider the introduction of a meta-model as a central contribution to improve the completeness and correctness of the security requirement engineering process described in [AR13]. To reach those objectives, we leverage the following capabilities of the knowledge-centric meta-model:

1. **Agreement on Definitions**. Different stakeholders (system engineers, risk experts, security experts, verification and testing teams, etc.) are involved in the system design and development lifecycle. The specification of terms and principles of IT security in a common knowledge base within a given requirements engineering process is known to be very beneficial to security [ISO09a]. Security terms and definitions from security standards (e.g., ISO/IEC 17799:2005, ISO/IEC 27002:2005, etc.) can help build a common knowledge base to bridge a potential miscommunication gap.

2. **Elicitation of Requirements**. In addition to security concepts definitions, the meta-model also describes relationships between such concepts. Such relationships

can be used to improve the elicitation of requirements. For instance, the IEEE 830 standard recommends to look after eight characteristics in order to achieve good software requirements specifications, out of which four (unambiguity, correctness, consistency, and completeness) are related to the semantics and documentation of a requirement specification. The use of specialization/generalization relationships between concepts together with domain-specific ontologies elaborated by experts obviously reduces the ambiguity and increases the correctness of a specification. It also can provide some assistance in the formalization of a use case by inciting the practitioner to author semantically richer models.

3. **Consistency checking**. Coverage relationships address completeness concerns: for instance, relating goals and attacks (anti goals) to identify requirements as done in KAOS ascertains that all threats are addressed or conversely, that all requirements address a problem. Finally, the availability of a semantic network of concepts and relationships makes it possible to check that an instance of a concept does not introduce contradictory semantics, thereby ensuring consistency through a more complex reasoning. Such a reasoning might also help address contractions between security and safety [PCB13].

4. **Prioritization of requirements**. The initial set of requirements can be organized into stakeholder-defined categories (e.g., essential, non-essential, etc.). One may use security standards and specifications to determine and categorize requirements together with functional requirements they reinforce. Nevertheless, we acknowledge the fact that, during security requirements prioritization, some of the requirements may be deemed to be unfeasible to implement. Security requirements often conflict and interact with other system requirements (functional or not). For instance, mandated security requirements might conflict with what can be done within a reasonable timeframe or budget. In this case, security engineers have the option to explicitly document the dismissal of requirements, or to label them for "future consideration".

2.3 Knowledge-centric meta-modeling

Our meta-model is organized in core concepts that underlie all security requirements engineering methodologies, even though they follow different workflows. The essential artifacts that we identified are: security goals, security attacks, system models (behavioral and structural), and use case oriented models. We consequently organize these artifacts under the form of security classes so that the security metadata produced at different phases of the security requirements engineering process can be easily shared and reused. Each class is described by a domain specific ontology, in a knowledge-centric manner. For example, security goal related metadata should be useable for identifying system assets. Similarly such metadata should be useable for analyzing security attacks and vulnerabilities. Figure 1 summarizes the ontology-driven security requirement engineering methodology.

3 Security ontologies

We define in this section the security ontologies discussed above. They have been modeled with the Ontology Web Language [DevH$^+$03] using OWL classes. Core concepts are (1)

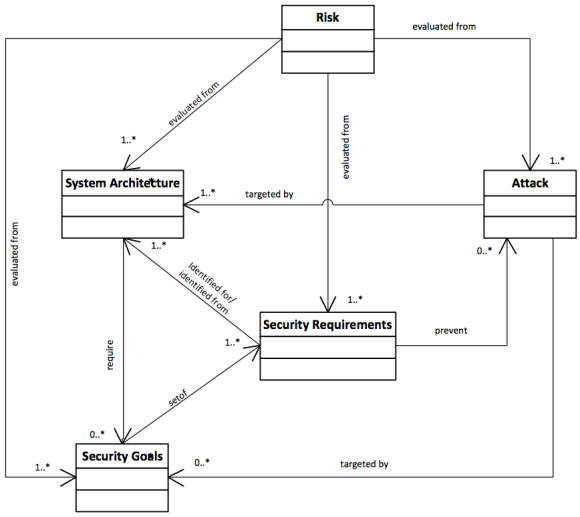

Fig. 1: An ontology-driven security requirement engineering methodology

security goals, (2) system architecture, (3) security attacks, (4) security requirements, and (5) security mechanisms[1]. Each security ontology constitutes a knowledge repository for capturing, classifying, and sharing security related information. With regards to our objectives, ontologies are used to structure and organize SysML-Sec artifacts, first of all through the introduction of a controlled vocabulary. The structure of our security ontologies is flexible and easily extendable, which makes the seamless addition of new concepts possible.

3.1 Ontology of attacks

Figure 2 summarizes our analysis regarding the concepts that can be excerpted from well-known security standards (ISO/IEC 18045, ISO/IEC 27000: 2012, ISO/IEC 17799:2005, NIST SP-800:30, etc.) and security dictionaries (CVE, CAPEC, OWASP, CLASP, etc.) in order to build the security attack ontology. The *Attack Type* depicts an attempt to destroy, expose, alter, disable, steal, or gain unauthorized access to make unauthorized use of system assets [ISO09b]. This abstract level of attack type definition is taken from e.g., NIST SP 800-30 ([NIS12], sec. 3.2). Attack types are then categorized into the Threats and Vulnerabilities sub classes: a *threat* is "potential cause of an unwanted incident, which may result in harm to a system or organization"; a *vulnerability* is "a mistake in software that can be directly used by a hacker to gain access to a system or network" [CVE]. *Attack Consequences* refer to the impact of a security breach or to outcomes that are not the ones intended by a purposeful system action. The attack consequences are classified into usurpation, disruption, deception, disclosure, etc. An *adversary* is a threat agent according to the ISO/IEC 21827:2008 standards ([ISO08], sec. 3.35), who attempts to attack system assets that have value to the stakeholders. An adversary may range from a very unskilled individual to an expert group. In order to anticipate and thwart

[1] In this paper, we only discuss attacks and requirements. Further details can be found in [Idr12]

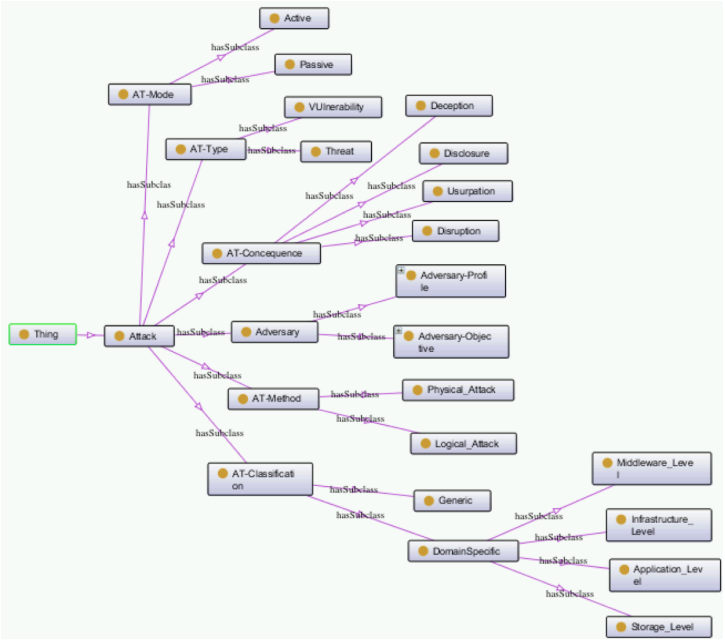

Fig. 2: Ontology of attacks

the expected types of attacks, one must have a solid understanding of the adversary's perspective and his/her capabilities and know-how about attack potential. Different attack objectives and corresponding adversary profiles can be considered. The *Attack Method* is related to the attack mode class. The attack method can be classified as either logical or physical. The *Attack Classification* class is defined to categorize and to systematically aggregate classes that precise the description of attacks and its objectives. A collection of criteria, including security dictionaries (i.e., CVE, CAPEC, OWASP, CLAP) can be used to determine and cluster security attacks and vulnerabilities. However, at the most abstract level, we can classify security attacks into generic attacks that target languages and execution environments, for instance, and domain specific attacks.

3.2 Ontology of security requirements

The ontology of security requirements focuses on the different constructs and concepts defined in security requirements specifications (for instance KAOS, UMLsec, or TTool) and security standards. This ontology aims to detect the missing security construct in security requirements frameworks and to enrich their semantics. The core classes and the concepts identified for the security requirements ontology are *Functional Security Requirements*, *Non-Functional Security Requirements*, *Classification* (e.g., generic, domain-dependent), *Specification* (formal, semi-formal, informal), *Assumptions* (e.g., related trade-offs), and *Role* (individuals and/or teams involved in the definition of requirements). We also include the *Relationship* class which depicts SysML-Sec relationships, like *refine*, *derive*, *copy*, *containment*, *verify*, and *trace*. The ontology of security requirements also makes it possible

to describe traditional security objectives such as confidentiality or integrity as important concepts.

4 Integrating Knowledge Bases and Reasoning into SysML-Sec

Syntactically, SysML and ontology languages (i.e., OWL, OIL, etc.) have a lot of similarities. While SysML makes use of a graphical formalism, it also aims at defining the semantics of a system with constructs like blocks, associations, part properties, and relationships between models and sets of model elements. Ontology languages use classes, properties, relationships, and individuals as basic knowledge constructs. For instance, OWL defines classes by appropriate and implicit logical constraints on properties of their subclasses and concepts. The integration of both approaches enables engineers to add reasoning arguments to the explicit documentation of system models, and to define more precise relationships in the course of a typical model-based development process. We discuss in the following how ontologies and inferences on ontologies are used to enrich the SysML-Sec framework.

4.1 SysML Diagrams and Ontological concepts

We prototyped the introduction of security concepts from our ontologies into SysML-Sec models. SysML-Sec diagrams are firstly annotated with ontological concepts that thus establish a mapping between SysML stereotypes or relationships and OWL classes and relationships. Secondly, a number of attributes are also extended through the use of a controlled vocabulary defined by the ontology instead of free text. For instance, the security requirement and security attack ontologies can be used to precise the semantics of the security requirement and attack tree diagrams, respectively.

For example, Figure 3 depicts the extension of the SysML-Sec modeling capabilities with those two approaches. We have defined the $<< attack >>$ stereotype to represent the security attack concept of the ontology in the "part" element. The properties of the "part" element shown here in (a) are concepts and terms defined in the security attack ontology shown in (b). The different classes defined in the ontology of security attacks are mapped to a corresponding property name. Instances of ontology classes in relationship with an instance of one such class can be used to further tag the SysML-Sec diagram, in particular through attributes of a SysML-Sec block. Here for instance, we make use of a controlled vocabulary describing the adversary as either a layman, an expert, or a professional attackers.

4.2 Reasoning with SysMLsec Models

The ontologies can be used to reason about the security concepts defined in SysML-Sec models. In particular, our objective is to enable the security engineers to have access to various ontological concepts and their relationships, and to reason over such semantic graphs. The integration of these capabilities thus relies on the introduction of a transformation engine to bridge the two approaches, as depicted in Figure 4. We prototyped such a system in order to perform consistency checks about the coverage of attacks and security goals as defined in the SysML-Sec parametric diagram and requirement diagram respectively. These checks can be written as SPARQL queries once the translation engine has created class instances out of the content of the SysML-Sec diagrams.

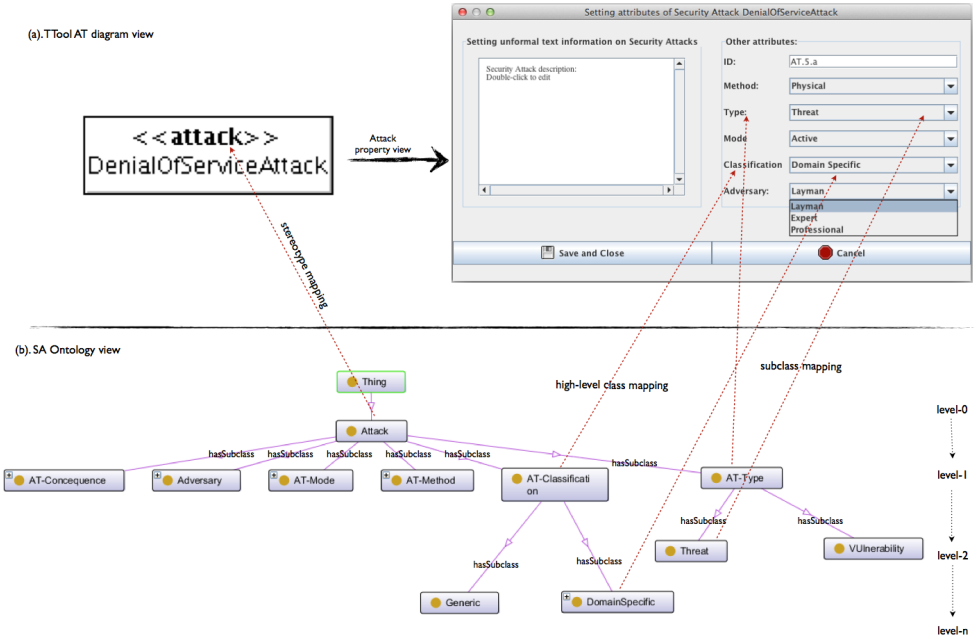

Fig. 3: Mapping of the security attack ontology concepts to the SysML-Sec attack tree diagram and attributes (depicted within TTool supporting dialogs)

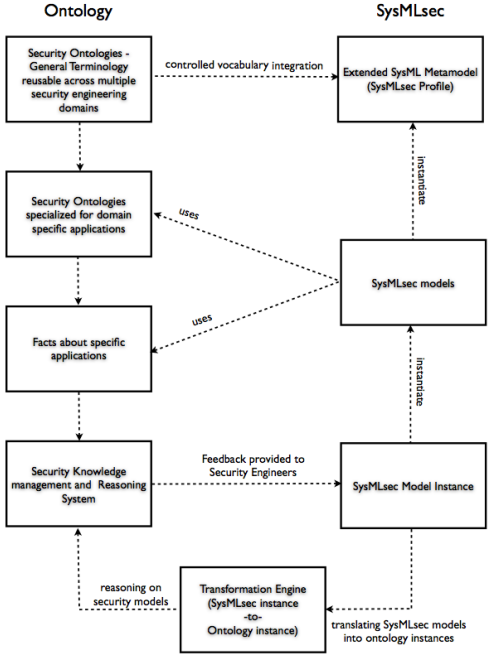

Fig. 4: Integration of security ontologies and reasoning into SysML

5 Related work

Ontologies have been developed in various contexts of security engineering, notably: for explaining threat and security mechanisms [ALRL04], for risk assessment [EFKW07], security management [TG06], security protocol designs [KLK05], policy configuration [BSL+11], and security requirements [KBD+06] [HLMN08], among others. They have also been used to find out about discrepancies and incompletenesses in security standards [VFZL10]. The use of ontologies for the precise expression of requirements has been studied for a long time [LFB96]. Building specifications or documentation is seemingly the most commonly evaluated application of ontologies [GKM09]. Ontologies also bring clear semantics to concepts and relationships between different security artifacts as pointed out by [SBO07]. The use of ontologies for checking requirement consistency as well as for their traceability is probably one of the areas that have attracted most attention so far [STZ+11] [LFB96]. Siegemund et al. [STZ+11] for instance describe how to check the consistency and completeness of goal-oriented requirement specifications by combining ontology consistency checking and rule driven completeness checks. Similarly, Lin et al. [LFB96] described an ontology driven solution for generating unambiguous and precise requirement specification that can be easily extendable and support dependencies and relationships among requirements. Cranefield [Cra01] describes the use of ontological reasoning to extract knowledge from UML models. Ontologies can also be used to share and reuse requirement related knowledge [DRR+05] with other models that are relevant to requirements. The integration of reasoning and ontological concepts into SysML has recently become a hot topic in the requirements engineering domain [Gra09, WBK+12].

6 Conclusion and future work

The integration of knowledge representation techniques, in particular ontologies, into model-driven engineering tools may increase the accuracy and detail of specifications we capture, especially in the field of security requirements engineering. An important outcome of the integration of such techniques will be the capability to also capture inferences over ontological concepts and relationships, that is the ability to reason over knowledge. This capability opens up the door to the automation of checks on the consistency and completeness of specifications performed by hand only today. We implemented a proof-of-concept prototype of such a system based on the OWL language. We also envision the possibility to eventually express formal specifications of best practices of secure system design through the implementation of meta-model knowledge inferences rules. Although our tools and methodology [AR13] are dedicated to embedded systems and focuses on hardware/software mapping concerns, our proposal to integrate knowledge representation techniques with a requirements engineering process can likely be adapted to more general purpose system architectures like Service Oriented Architectures. We are currently in the process of developing our prototype to integrate it more effectively in the SysML-Sec framework in order to further experiment with its capabilities.

References

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure*

Computing, IEEE Transactions on, 1(1):11–33, Jan 2004.

[AR13] L. Apvrille and Y. Roudier. SysML-Sec: A Model-Driven Environment for Developing Secure Embedded Systems. In *Proc. of SARSSI 2013, Mont-de-Marsan, France*, September 2013.

[BSL+11] C. Basile, J. Silvestro, A. Lioy, D. Canavese, M. Arrigoni Neri, S. Paraboschi, M. Casalino M. Verdicchio, and T. Scholte. Security Ontology Definition. In *Technical Report D3.2, PoSecCO Project*, 2011.

[Cra01] S. Cranefield. UML and the Semantic Web. In *In Proc. of the International Semantic Web Working Symposium*, 2001.

[CVE] CVE. Common Vulnerabilities and Exposures. http://cve.mitre.org/.

[DevH+03] M. Dean, G. Schreiber (eds.), F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL Web Ontology Language Reference. 2003.

[DRR+05] Bjorn Decker, Eric Ras, Jorg Rech, Bertin Klein, and Christian Hoecht. Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. In *In Workshop on Semantic Web Enabled Software Engineering (SWESE)*, 2005.

[EFKW07] A. Ekelhart, S. Fenz, M. Klemen, and E. Weippl. Security Ontologies: Improving Quantitative Risk Analysis. In *Proc. of the 40th Annual Hawaii International Conference on System Sciences, HICSS'07*, page 156, January 2007.

[FGH+10] Benjamin Fabian, Seda Gürses, Maritta Heisel, Thomas Santen, and Holger Schmidt. A comparison of security requirements engineering methods. *Requirements Engineering Journal, Special Issue âĂŞ Security Requirements Engineering*, 15(1):7–40, March 2010.

[GKM09] D. Gasevic, N. Kaviani, and M. Milanovic. Ontologies and Software Engineering. In *In Handbook on Ontologies, Springer*, pages 593–615, 2009.

[Gra09] H. Graves. Integrating SysML and OWL. In *OWL Experiences and Directions October Workshop, OWLED'09*, 2009.

[HLMN08] Charles B. Haley, Robin Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Security Requirements Engineering: a Framework for Representation and Analysis. *IEEE Transactions on Software Engineering*, 34(1):2008, 2008.

[Idr12] Muhammad Sabir Idrees. *A Requirements Engineering Driven Approach to Security Architecture Design for Distributed Embedded Systems*. PhD thesis, Telecom ParisTech, 2012.

[ISO08] ISO/IEC-21827:2008. Information Technology - Security Techniques - Systems Security Engineering - Capability Maturity Model SSE-CMM, 2008.

[ISO09a] ISO/IEC-15408:2009. Information Technology - Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model, 2009.

[ISO09b] ISO/IEC-27000:2012. Information Technology - Security Techniques - Information Security Management Systems - Overview and Vocabulary, 2009.

[KBD+06] M. Karyda, T. Balopoulos, S. Dritsas, L. Gymnopoulos, S. Kokolakis, C. Lambrinoudakis, and S. Gritzalis. An ontology for secure e-government applications. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, April 2006.

[KLK05] Anya Kim, Jim Luo, and Myong Kang. Security ontology for annotating resources. In *Research Lab, NRL Memorandum Report*, page 51, 2005.

[LFB96] Jinxin Lin, Mark S. Fox, and Taner Bilgic. A Requirement Ontology for Engineering Design. *Concurrent Engineering: Research and Applications*, 4:279–291, 1996.

[MBSFM10] Daniel Mellado, Carlos Blanco, Luis E. SÃąnchez, and Eduardo FernÃąndez-Medina. A systematic review of security requirements engineering. *Computer Standards & Interfaces*, 32(4):153 – 165, 2010.

[NIS12] NIST-SP-800:30. Risk Management Guide for Information Technology Systems., September 2012.

[NNY10] Armstrong Nhlabatsi, Bashar Nuseibeh, and Yijun Yu. Security Requirements Engineering for Evolving Software Systems: a Survey. Technical Report 1, The Open University, 2010.

[PCB13] L. Pietre-Cambacedes and M. Bouissou. Cross-fertilization between safety and security engineering. *Rel. Eng. & Sys. Safety*, 110:110–126, 2013.

[SBO07] Rainer Schmidt, Christian Bartsch, and Roy Oberhauser. Ontology-based Representation of Compliance Requirements for Service Processes, 2007.

[STZ+11] K. Siegemund, E. J Thomas, Y. Zhao, J. Pan, and U. Assmann. Towards Ontology-driven Requirements Engineering. In *7th International Workshop on Semantic Web Enabled Software Engineering*, 2011.

[TG06] B. Tsoumas and D. Gritzalis. Towards an Ontology-based Security Management. In *Proc. of the 20th International Conference on Advanced Information Networking and Applications, AINA'06*, pages 985–992, 2006.

[VFZL10] V.I. Vorobiev, L.N. Fedorchenko, V.P. Zabolotsky, and A.V. Lyubimov. Ontology-based analysis of information security standards and capabilities for their harmonization. In *Proc. of the 3rd International Conference on Security of Information and Networks, SIN'10*, pages 137–141, 2010.

[WBK+12] D.A. Wagner, M.B. Bennett, R. Karban, N. Rouquette, S. Jenkins, and M. Ingham. An Ontology for State Analysis: Formalizing the Mapping to SysML. In *Aerospace Conference, 2012 IEEE*, pages 1–16, March 2012.