The Case for Functional Security Requirements:
Deriving a Framework for Functional Security Requirements Engineering

Bryan Robbins
Department of Computer Science and Engineering
Mississippi State University
Starkville, MS, USA

Abstract - *The following paper investigates the integration of requirements engineering and security engineering concepts in search of an effective process for functional security requirements engineering. A particular focus is placed on the end goal of deriving assurance in the system's capabilities in the scope of a full lifecycle requirements engineering effort. The development of an integrated framework is supported by a review of current literature and sound principles of security and requirements engineering. The framework is presented in an incremental fashion, as a basic requirements engineering process is transformed into a requirements engineering process which considers functional security requirements at each phase.*

Keywords: *requirements engineering, functional security requirements*

1 Introduction

Many paradigms exist for the consideration of security requirements in software systems. At the requirements phase of the software engineering life cycle, consideration of security falls into two classes. Some [4] consider security a chief nonfunctional requirement, typically delaying its consideration until more detailed levels of the engineering process, such as the software design. A functional requirement-based approach is advocated by [5, 6], but comes at a cost.

These two options need not stand in direct opposition to one another. In fact, it is inarguable that security considerations cannot be narrowed down to a numerable list of low-level statements nor a likeminded list of high-level goals – neither would be effective alone. With this dichotomy in mind, the following paper will investigate an optimal process for the consideration of security from a functional perspective during requirements engineering (RE) by integrating a model of RE which spans the software lifecycle with principles from security engineering, claiming that security functional requirements (SFRs) can be properly engineered to provide traceability of security mechanisms throughout the entire development lifecycle, and examination of the artifacts of traceability can be coupled with the proper identification of security requirements to allow for the assumption of some level of assurance in the correct functionality of these mechanisms.

Security requirements engineering (SRE) can be considered the intersection of security engineering techniques with the software RE process. Let us begin our discussion of SRE with a look at a prototypical security engineering process, then investigate how that process and its principles can be integrated with a sound requirements engineering process.

2 Security Engineering

Security engineering concerns itself with assuring that qualities of specific system entities are met. The entities referenced can include data, presumably in the form of files within a filesystem, and also active entities such as processes. A number of sources, including [5], identify the qualities associated with security as confidentiality, integrity, and availability. From a DoD standpoint, the discussion regarding the attributes that computer security must protect was initiated by [14]. Most sources include these three fundamental qualities, with some [7] adding additional related qualities such as accountability (or non-repudiation). In summary, efforts to secure a system involve providing the qualities of confidentiality, integrity, availability, and accountability to both its active and passive entities.

There are a number of methods followed in the practice of security engineering in order to consider these qualities. A traditional approach to security engineering would be consideration of a policy (or in fact a number of policies) that must be enforced regardless of the state of the system. A prototypical method for such enforcement of policy is described by [1] as a layered method of increasingly specific descriptions of what enforcement of a higher level policy should look like, thus mapping a policy as general as, “Protect the company’s information assets” to the particular mechanisms which will enforce the original policy. Abrams et al. note that this transcendence from policy to mechanism mirrors that of the TCSEC paradigm described in [7].

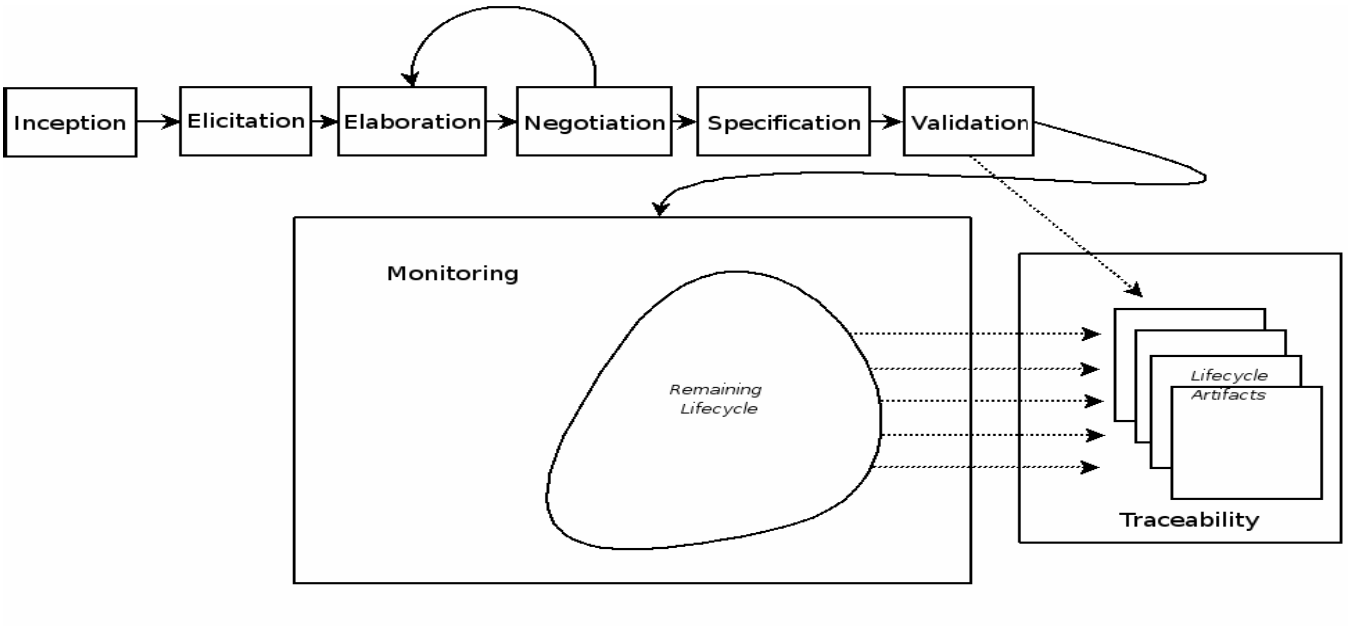

Figure 1: Basic Requirements Engineering (RE) Process

In practice, so that adherence to the policy is assured, security policies are often mapped to formal mathematical models of security, proven mathematically to provide the desired security qualities if precisely implemented. For the sake of example, such a model is the Bell-LaPadula (BLP) model of confidentiality, originally described in [3], which describes a way in which the access of objects (passive entities of the system) by subjects (active entities of the system) can be controlled. The enforcement of the BLP model's underlying confidentiality policy is a constraint that the security engineering process must integrate into the system.

Using the approach of Abrams et al. in [1] with regard to BLP, this would include a series of increasingly specific system constraints, which draw initially on the four necessary properties of the BLP model, and digress to the "trusted subjects" identified by [3] necessary to enforce the properties, and furthermore to the implementation of these subjects. This is an example of security engineering as a standalone process. Because such an approach is analogous to top-down requirements engineering, let us now consider the integration of these two fields.

3 Requirements Engineering

The process of requirements engineering (RE) is often divided into a varying number of phases. Several sequential RE phases are enumerated by Pressman in his definitive software engineering text [12] to be: inception, elicitation, elaboration, negotiation, specification, and validation. Additionally, the RE process also includes umbrella activities such as requirements monitoring and traceability that we will also consider, as depicted in Figure 1.

In Figure 1 the bold arrows indicate the flow of work between phases while dashed arrows indicate a typical documentation trail throughout the lifecycle, which will be significant in the latter two phases of the process, monitoring and traceability. We will call these two "RE phases" although their scope is somewhat broader than that of previous phases and their execution parallel rather than sequential. Both are addressed not in the requirements phase of the software lifecycle, but throughout the remaining lifecycle. Taking a look at the phases of the RE process will allow us to carefully consider functional security requirement concerns in search of an integrated approach to security requirements engineering.

3.1 Inception and Elicitation

Inception involves the identification of system stakeholders and basic system characteristics. In the case of SRE, the process of selecting stakeholders would be affected by the need to include those with security engineering expertise who bring an entirely new set of stakeholder concerns to the process. These concerns are then expressed through the direct user feedback gathered during requirements elicitation. We would expect such concerns to parallel the goals of security engineering, such as maintaining the confidentiality, integrity, authentication, and accountability of system entities. These concerns may be expressed in the form of policies, whether existing or new, but at a high level. Such statements should avoid identifying mechanisms and focus on identifying concerns. This early in the RE process, it is important to gather as much stakeholder input as possible, particularly from security stakeholders, even if that feedback seems to have conflicting interests or foreseeable issues. Resolving these issues by molding security concerns into satisfactory requirements is part of the job of the next two phases of RE.

The consideration of security concerns at the inception and elicitation phases of RE requires no

additional phases or methods, simply expanding existing RE techniques such as stakeholder identification and initial stakeholder concern elicitation to include security-minded personnel and their concerns of confidentiality, integrity, availability, and accountability.

3.2 Elaboration and Negotiation

The elaboration phase of RE has the goal of clarifying stakeholder ideas for the sake of sufficiently complete discussion. The resulting discussion is therefore carried out in the negotiation phase, as system developers begin to have a say in the requirements elicitation process, particularly in matters of technical feasibility. Negotiation can be considered a modification of the elicitation phase, however, it is unique from the initial elicitation phase in that system developers have become more involved and feedback is more focused and structured. These phases alternate until a satisfactory statement of requirements is reached, and the requirements are ready for specification. These are two of the most involved phases of RE, and consequently, require much discussion regarding SFR consideration.

3.2.1 Elaboration Frameworks

A structured statement of security concerns must be generated for use in these phases, and many have presented methods for accomplishing this task. A review of current literature in security RE will allow the abstraction of security requirements elaboration techniques and the integration of these newly identified phases with the existing RE framework of figure 1.

The security RE framework of Haley et al. uses a threat description method to enumerate a set of "preventative security goals" for a system, as presented in [8, 9, 10], although their research assigns security requirements as constraints on traditional FRs rather than FRs themselves. Alexander, in [2], holds that security is a nonfunctional requirement, but proposes the use of a threat-oriented technique known as misuse cases to elicit security requirements. [5] describes a threat description process in which security threats are identified as a three-way relationship between agents, actions, and assets that must eventually be addressed by countermeasures. These countermeasures are then mapped indirectly to the functional security requirements of [6]. Finally, van Lamsweerde proposes a goal/antigoal combined framework for the purpose of requirements elaboration in [13], using antigoals as a stimulus for threat consideration, threat identification and subsequent security requirements elicitation via a "threat tree".

This list, while certainly not exhaustive, gives us an idea of the shape that the statement of security requirements can take. To generalize, we see that some frameworks explicitly state "positive" security goals [10, 13] first, while others begin with threat analysis [2, 5]. However, of those identified above, all use threat analysis as a means to security requirements identification, and all generate positively testable requirements as their end product, which is essential to the RE process.

3.2.2 Generalizing Elaboration Frameworks

We can therefore create a security requirements elaboration framework that includes phases of goal identification, threat identification, countermeasure identification, and countermeasure modeling. At the countermeasure modeling phase, we can synchronize once again with the generic RE framework, as we have sufficiently bridged the gap between the objective statement of security goals and the ability to model (or otherwise sufficiently state) these goals as FRs. The new SFR elaboration phases are consistent with an iterative, policy-driven security engineering technique such as that of [1] as they support the iteration of elaboration and negotiation phases until an agreement is reached. The introduction of security policies and even formal models as discussed earlier is appropriate in the countermeasure modeling phase. Integration of these newly identified phases with the generic RE framework is shown in Figure 2, with the four new phases outlined.

Observe that the SFR elaboration phases run parallel to the standard elaboration phase, so that only SFRs are considered under the new phases and then rejoin the existing framework for the negotiation phase. When requirements have been sufficiently addressed, the next phases of RE begin to transition into development of the system.

3.3 Specification and Validation

The specification phase of RE is one of the most familiar in the process because it actually produces an artifact to be used in later phases. Many RE efforts use a template such as [11], which describes the IEEE format for software requirements specifications, as the means and the end of their RE process. When the time comes to specify an appropriately negotiated FR in an artifact, a structured approach such as that provided by [11] is preferred. SFRs, though not in wide use today, can be enumerated within an SRS if an integrated approach is used, but should likely receive special distinction due to the special consideration they will receive later in the lifecycle. The approach of [5] is to include SFRs, as a subset of the predefined countermeasures of [6] in a special document known as the "security target" that will guide evaluation procedures (which are the central purpose of the Common Criteria approach). A separate document may often be beneficial; however, it suggests separate methods for SFR engineering, which is not consistent with the integrated

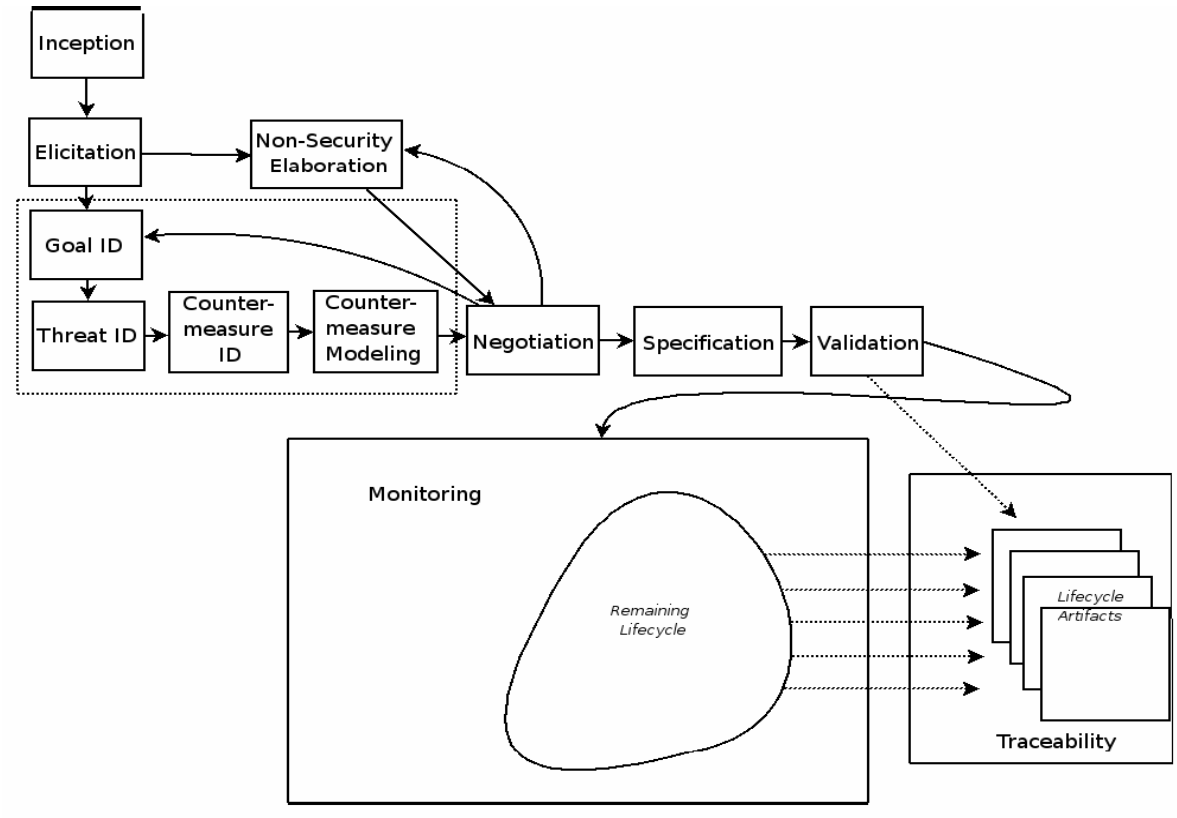

Figure 2: RE Process with Security Elaboration Phases

approach described thus far, and lends itself to gaps in traceability and difficulty in monitoring. However, it is conceivable that a separate evaluation-oriented document (such as the Common Criteria's security target) could be generated as a subset of an integrated software requirements specification.

After being stated in their negotiated form, all requirements (including SFRs) should be checked against a list of attributes to serve as validation. More formal methods also exist for validation of requirements. The primary purpose of this phase is to solidify traceability within the requirement specification. When SFRs are being validated, the attribute of testability is of particular interest. If the phases outlined above are followed for SFRs, statements such as “The system shall not ...”, which are by nature untestable, should have been converted into statements similar to “The system shall provide countermeasure X to ensure attribute Y on entity Z”. Similar attributes of traceability should also be ensured by the process derived above. Consequently, no additional phases are necessary in the integrated RE framework for the consideration of SFRs. As the software lifecycle is now ready to proceed past the requirements phase, let us consider SFR issues in the two remaining phases of RE.

3.4 Monitoring

Requirements monitoring is an important issue that must be part of an integrated RE approach, particularly when security is a chief concern. Essentially, we can state that properly engineered requirements, such as those generated during the first phases of the integrated framework presented above, are rendered ineffective if not properly monitored. It is understood that software projects, even in linear development models, have a tendency to change as development proceeds. Particularly with regard to SFRs, we need a mechanism for reevaluating security concerns upon each requirement insertion or modification. In fact, reevaluation of security may be deemed necessary based on architectural or design decisions that may not directly affect the specification of non-security requirements. We must ensure that the previously identified goals, threats, and countermeasures still adequately address security concerns in the modified system. To do so, we will add a conditional loop back to the goal identification phase of RE so that such considerations can be made upon requirements changes or other security-relevant modifications to the system. Note the presence of this loop in Figure 3. Requirements monitoring is a maintenance activity that should receive significant attention during development, but also be considered in planning efforts and as part of a quality

assurance methodology to minimize the cost of this

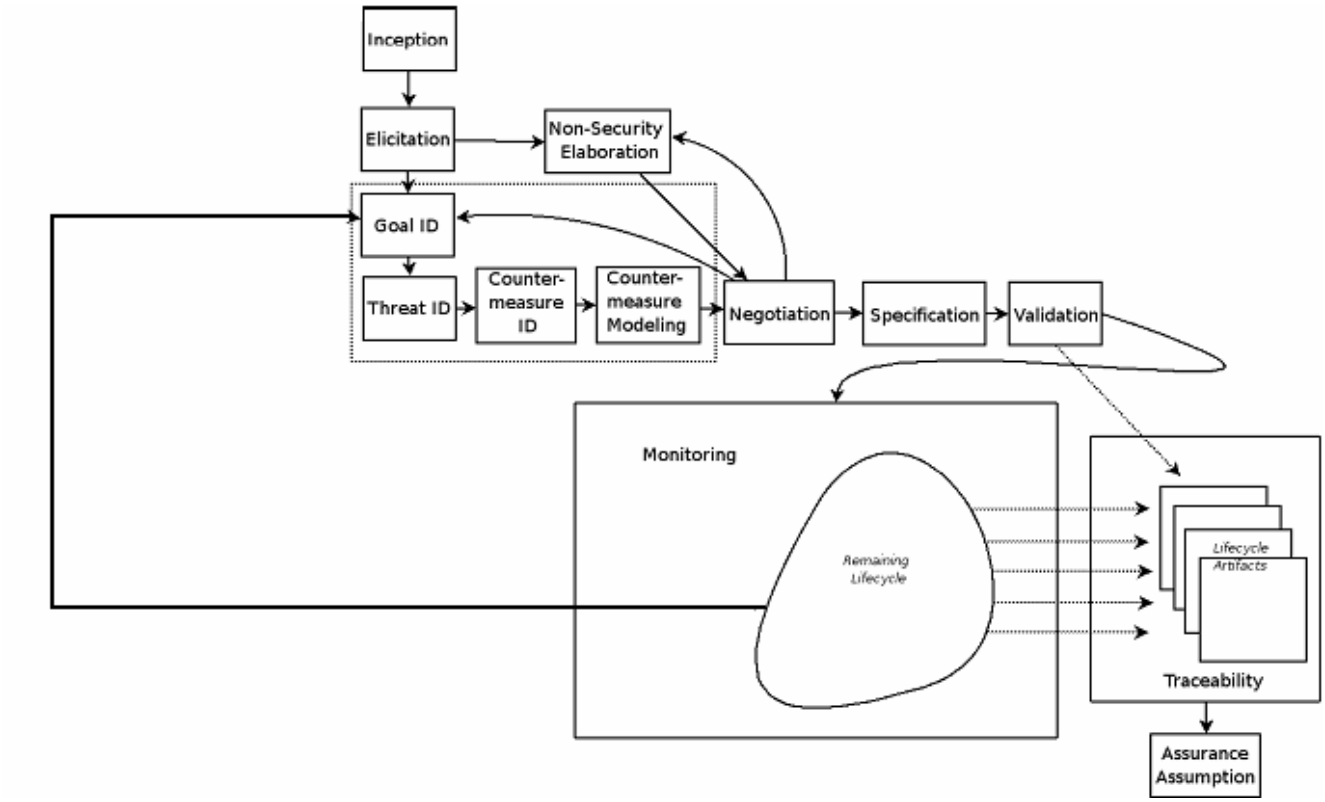

Figure 3: Integrated RE Process for Security Functional Requirements

potentially expensive activity.

3.5 Traceability

Traceability is nearly native to properly engineered functional requirements. Throughout our SFR integration process, we have mentioned traceability as a motivator for clean integration. The primary reason for this is the goal of assurance in system security. The Common Criteria methods of [5] are designed to eventually assign an assurance level to a product through an evaluation procedure. In the assignment of assurance levels, traceability is the requirement that increases, as it is traceability in development that provides us with the evidence we need in order to make assumptions regarding the system's performance, particularly in terms of security. Before releasing a product, assurance of security should be derived based on the completeness of SFRs and their traceability throughout the lifecycle. This is advocated by [5], as mentioned, and also by [8, 10] which support the analysis of requirement satisfaction at or near the end of the lifecycle, and should be considered the final phase of RE. Note the addition of an assurance assessment phase in Figure 3.

4 Conclusions

Considering the benefits of functional requirements, this paper has attempted to integrate security engineering concepts and existing frameworks from current literature into a basic RE model for functional requirements. The inception and elicitation phases were left untouched, but it was observed that security stakeholders and their concerns should be included in the activities of these phases. The elaboration phase was expanded for SFRs by abstracting out the phases of existing security elaboration frameworks to add four additional phases to the RE model which bridge the gap between the statement of high-level security goals centered on the confidentiality, integrity, availability, and accountability of system entities and a structured statement (perhaps via a policy or formal model) of countermeasures that protect against threats to these goals.

This allows for the negotiation phase of the integrated RE model to be identical for SFRs and non-security FRs. The specification and validation phases were also left unchanged as specification and validation concerns should be addressed by the newly added SFR elaboration phases. However, specification of SFRs can be considered a subset of the specification of all FRs for the sake of preparing evaluation documentation. Requirements monitoring for SFRs necessitated the addition of a loop within the RE

model to reevaluate security provisions upon significant changes to the system. The superior traceability of FRs allowed for the addition of an assurance assumption phase that assesses the confidence level of security measures in a system, capturing one of the primary goals of effective security engineering. Through these mechanisms, an effective integration of RE with security engineering has been achieved. As this work is currently purely theoretical, future work should include application of this integrated framework to case studies and applicable software development exercises.

References

[1] M. D. Abrams, S. Jajodia, and H. J. Podell, "Security Engineering", *Information Security: An Integrated Collection of Essays*, IEEE CS Press, Los Alamitos, California, USA. pp. 330-349.

[2] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent", *IEEE Software*, Jan 2003.

[3] D. E. Bell, L. J. La Padula, "Secure Computer System: Unified Exposition and Multics Interpretation", March 1976.

[4] L. Chung, B. A. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach", *International Conference on Software Engineering*, Seattle, Washington, USA, 1995.

[5] *Common Criteria for Information Technology Security Evaluation*, "Part 1: Introduction and General Model", September 2006, Version 3.1, rev. 1.

[6] *Common Criteria for Information Technology Security Evaluation*, "Part 2: Security Functional Components", September 2006, Version 3.1, rev. 1.

[7] Department of Defense Standard, "Department of Defense Trusted Computer System Evaluation Criteria", DOD 5200.28-STD, December 26, 1985.

[8] C. B. Haley, J. D. Moffett, R. Laney, B. Nuseibeh, "Arguing Security: Validating Security Requirements using Structured Argumentation", *Proceedings of Third Symposium on Requirements Engineering for Information Security*, Paris, France, August 29, 2005.

[9] C. B. Haley, J. D. Moffett, R. Laney, B. Nuseibeh, "Deriving Security Requirements from Crosscutting Threat Descriptions", *Proceedings of Third International Conference on Aspect-Oriented Software Development*, Lancaster, UK, ACM Press, March 22-26, 2004, pp. 112-121.

[10] C. B. Haley, J. D. Moffett, R. Laney, B. Nuseibeh, "A Framework for Security Requirements Engineering", *Proceedings of 2006 Software Engineering for Secure Systems Workshop*, Shanghai, China, May 20-21, 2006, pp. 35-42.

[11] IEEE Std. 830-1998, *Recommended Practice for Software Requirements Specifications*, IEEE, New York, NY, June 25, 1998.

[12] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGrawHill, New York, 2005.

[13] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional AntiModels", *Proceedings of 26th International Conference on Software Engineering*, Edinburgh, Scotland, UK, May 23-28, 2004.

[14] W. Ware, ed. "Secure Controls for Computer Systems: Report of the Defense Science Board Task Force on Computer Security", *RAND Corporation*, Santa Monica, CA, February 11, 1970.