Software Engineering Institute | Carnegie Mellon University

Requirements Engineering Annotated Bibliography

Nancy Mead

January 2006

BIBLIOGRAPHY

Abrams, M. D. & Brusil, P. J. “Application of the Common Criteria to a System: A Real-World Example.” Computer Security Journal 16, 2 (Spring 2000): 11-21.

A new paradigm for specifying and evaluating information system security is on the verge of international standardization. Early use of these standards for specifying and acquiring large distributed systems is described. Unresolved issues concerning commercial-off-the-shelf component product integration and system evolution are addressed.

Alberts, C. & Dorofee, A. Managing Information Security Risks: The OCTAVE Approach. New York: Addison Wesley, 2003.

This is a descriptive and process-oriented book on a new security risk evaluation method, OCTAVE. OCTAVE stands for Operationally Critical Threat, Asset, and Vulnerability Evaluation "^{SM}." An information security risk evaluation helps organizations evaluate organizational practice as well as the installed technology base and to make decisions based on potential impact.

OCTAVE^{SM} enables any organization to develop security priorities based on the organization's particular business concerns. This approach provides a coherent framework for aligning security actions with overall objectives. Managing Information Security Risks, written by the developers of OCTAVE, is the complete and authoritative guide to its principles and implementations. The book provides a systematic way to evaluate and manage information security risks, illustrates the implementation of self-directed evaluations, and shows how to tailor evaluation methods to different types of organizations.

Alexander, I. “Misuse Cases: Use Cases with Hostile Intent.” IEEE Software 20, 1 (January-February 2003): 58-66.

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

Humans have analyzed negative scenarios ever since they first sat around Ice Age campfires debating the dangers of catching a woolly rhinoceros: "What if it turns and charges us before it falls into the pit?" A more recent scenario is "What if the hackers launch a denial-of-service attack?" Modern systems engineers can employ a misuse case, the negative form of a use case, to document and analyze such scenarios. A misuse case is simply a use case from the point of view of an actor hostile to the system under design. Misuse cases have many possible applications and interact with use cases in interesting and helpful ways. The paper discusses the elicitation of safety requirements from failure cases and considers the interplay of design, functional, and nonfunctional requirements.

Allen, J. H.; Barnum, S.; Ellison, R. J.; McGraw, G.; & Mead, N. R. Software Security Engineering: A Guide for Project Managers. Boston, MA: Addison-Wesley Professional, 2008 (ISBN: 978-0321509178).

Software that is developed from the beginning with security in mind will resist, tolerate, and recover from attacks more effectively than would otherwise be possible. While there may be no silver bullet for security, there are practices that project managers will find beneficial. With this management guide, you can select from a number of sound practices likely to increase the security and dependability of your software, both during its development and subsequently in its operation.

Berber, M.; von Solms, R.; & Overbeek, P. "Formalizing Information Security Requirements." Information Management and Computer Security 9, 1 (2001): 32-37.

Risk analysis, concentrating on assets, threats and vulnerabilities, used to play a major role in helping to identify the most effective set of security controls to protect information technology resources. To successfully protect information, the security controls must not only protect the infrastructure, but also instill and enforce certain security properties in the information resources. To accomplish this, a more modern top-down approach is called for today, where security requirements driven by business needs dictate the level of protection required.

Bharadwaj, R. "How to Fake a Rational Design Process Using the SCR Method," 3-4. SEHAS'03 International Workshop on Software Engineering for High Assurance Systems. Portland, OR, May 9-10, 2003. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, 2003. http://www.sei.cmu.edu/community/sehas-workshop/.

We explore the idea of faking a rational design process…by the application of the extended SCR Method of Heitmeyer and Bharadwaj. We argue that the formal artefacts created as a result serve as the basis for determining the work products associated with each step of the process, and whose quality assessment is aided by the application of tools in the SCR Toolset. Further, since the products associated with each step have a consistent formal denotation, the approach opens the possibility of significantly automating many process steps.

Biskup, J. & Bonatti, P. A. "Lying Versus Refusal for Known Potential Secrets." Data and Knowledge Engineering 38 (2001): 199-222.

Security policies and the corresponding enforcement mechanisms may have to deal with the logical consequences of the data encoded in information systems. Users may apply background knowledge about the application domain and about the system to infer more information than is explicitly returned as answers to their queries. Some of the approaches to dealing with such a scenario are dynamic. For each query, the correct answer is first judged by some censor and then—if necessary—appropriately modified to preserve security. In this paper, we contribute to the formal study of such approaches by extending the comparison of the two possible answer modifications (namely, lying and refusal) to the case of known potential secrets. First, we explicitly define the security requirements. Second, we extend to such requirements a previous results on security preservation using lies. Then we introduce a variant of the refusal-based approach that is suitable for potential secrets. Finally, we extensively analyse and compare the two approaches. We prove formally that, in general, they are incomparable in many respects, but, under fairly natural assumptions, lies and refusals lead to surprisingly similar behaviors and convey exactly the same information to the user. The latter result leads to a fundamental new insight on the relative benefits of the two approaches.

Biskup, J.; Flegel, U.; & Karabulut, Y. "Secure Mediations: Requirements and Design," 127-140. Database Security XII: Status and Prospects. Edited by S. Jajodia. Twelfth International Working Conference on Database Security, Chalkidiki, Greece, July 15-17, 1998. Norwell, MA: Kluwer Academic Publishers, 1999 (ISBN 0792384881).

We discuss the security requirements for mediation, and present our approach towards satisfying them, with an emphasis on confidentiality and authenticity. Furthermore we outline the design of the basic security mechanisms for mediators. Our basic approach suitably combines the concepts of credentials, for authentic authorization with some kind of anonymity, and of asymmetric encryption, for confidentiality, and it can be extended to include additional mechanisms like digital signatures and fingerprints. Additionally it adopts the model of role based security policies because of its application orientation and of its potentials to integrate and unify various policies.

Butler, S. A. & Fischbeck, P. "Multi-Attribute Risk Assessment." SREIS 2002, Second Symposium on Requirements Engineering for Information Security. Raleigh, NC, October 16, 2002. Lafayette, IN: CERIAS, Purdue University, 2002.

Best practice dictates that security requirements be based on risk assessments; however, simplistic risk assessments that result in lists of risks or sets of scenarios do not provide sufficient information to prioritize requirements when faced with resource constraints (e.g., time, money). Multi-attribute risk assessments provide a convenient framework for systematically developing quantitative risk assessments that the security manager can use to prioritize security requirements. This paper presents a multi-attribute risk assessment process and results from two industry case studies that used the process to identify and prioritize their risks.

Caplan, K. & Sanders, J. L. "Building an International Security Standard." IEEE IT Professional 1, 2 (March/April 1999): 29-34.

The Common Criteria for Information Technology Security Evaluation standard (CC) promises to replace scattered and often conflicting regional and national security standards. An emerging international standard, it is intended to support developers, evaluators and consumers of security products. The CC provides a framework to rate products by evaluation assurance level (EAL). Each EAL embodies a recommended set of assurance requirements: the higher the EAL, the more secure the product. You can use EALs to pick and choose which assurance requirements you want to satisfy. Think of the EALs as you would think of bandwidth or processor speed. Not everyone in your organization needs a dedicated T3 line or a 450 MHz desktop. Likewise, not every security product you use needs an EAL7 rating. The article shows how you, as a security products consumer, can use the CC to help determine if a given product meets your security needs. By rating different products according to their EALs, the CC can help you comparison shop and select an appropriately secure product. Further, the standard's international scope can help you integrate your system's security components with those in other countries-whether those components belong to customers, vendors, or other divisions of your own enterprise.

Caulkins, J.; Hough, E.; Mead, N. R.; & Osman, H. "Optimizing Investments in Security Countermeasures: A Practical Tool for Fixed Budgets." IEEE Security & Privacy 5, 5, 24-27.

Software engineers and businesses must make the difficult decision of how much of their budget to spend on software security mitigation for the applications and networks on which they depend. This article introduces a novel method of optimizing using integer programming (IP), the combination of security countermeasures to implement to maximize system security under fixed resources. The steps in the method and recent results with a case study client are described.

Chen, P.; Dean, M.; Lopez, L.; Mead, N. R.; Ojoko-Adams, D.; Osman, H.; & Xie, N. SQUARE Methodology: Case Study on Asset Management System (CMU/SEI-2004-SR-015). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04sr015.html.

This report exemplifies the application of the Systems Quality Requirements Engineering (SQUARE) methodology developed by the Software Engineering Institute's Networked Systems Survivability Program on an asset management application. An overview of the SQUARE process and the vendor is presented, followed by a description of the application under study. The nine-step process of requirements engineering is then explained, and feedback on its implementation is provided. The report concludes with a synopsis of the findings and recommendations for future work.

This report is one of a series of reports resulting from research conducted by the SQUARE Team as part of an independent research and development project of the Software Engineering Institute.

Chung, L.; Hung, F.; Hough, E.; & Ojoko-Adams, D. Security Quality Requirements Engineering (SQUARE): Case Study Phase III, CMU/SEI-2006-SR-003. Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/publications/documents/06.reports/06sr003.html

This special report is the third in a series focusing on the practical application of the Security Quality Requirements Engineering (SQUARE) process. In this report, a student team presents their results of working with three clients over the course of a semester. Each client was developing a large-scale software application and worked with the students to generate security requirements. With each client, the students implemented a different structured requirements-elicitation technique: Issue-Based Information Systems, Joint Application Development (JAD), and the Accelerated Requirements Method (ARM). In addition to an analysis of the three elicitation techniques, the student team also generated feedback and recommendations on different steps of the SQUARE process, such as requirements prioritization and inspection.

Clark, R. K.; Greenberg, I. B.; Boucher, P. K.; Lund, T. F.; Neumann, P. G.; Wells, D. M.; & Jenson, E. D. “Effects of Multilevel Security on Real-time Applications,” 120-129. Proceedings of the 9th Annual Computer Security Applications Conference. Orlando, FL, December 6-10, 1993. Los Alamitos, CA: IEEE Computer Society Press, 1993.

This paper presents a brief overview of a notional airborne application scenario that requires both multilevel security and real-time processing. It was used to guide decisions related to formation of the security policy interpretation, the operating system interface, and the system services design for a multilevel secure real-time distributed operating system (MLS RT DOS) called Secure Alpha. We compare secure and nonsecure application designs for the scenario to assess the effects of MLS RT DOS security features on the structure of the application. The comparison reveals that the security features make real-time scheduling and the construction of robust applications more difficult for this scenario.

Davis, Alan. Software Requirements: Objects, Functions, & States. Englewood Cliffs, N.J: Prentice-Hall Inc., 1993.

This revision of the bestselling software requirements book reflects the new way of categorizing software requirements techniques—objects, functions, and states. The author takes an analytical approach by helping the reader analyze which technique is best, rather than imposing one specific technique.

Ellison, R. J.; Fisher, D.; Linger, R. C.; Lipson, H. F.; Longstaff, T.; & Mead, N. R. Survivable Network Systems: An Emerging Discipline (CMU/SEI-97-TR-013, ADA341963). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.
http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html.

Society is growing increasingly dependent upon large-scale, highly distributed systems that operate in unbounded network environments. Unbounded networks, such as the Internet, have no central administrative control and no unified security policy. Furthermore, the number and nature of the nodes connected to such networks cannot be fully known. Despite the best efforts of security practitioners, no amount of system hardening can assure that a system that is connected to an unbounded network will be invulnerable to attack. The discipline of survivability can help ensure that such systems can deliver essential services and maintain essential properties such as integrity, confidentiality, and performance, despite the presence of intrusions. Unlike the traditional security measures that require central control or administration, survivability is intended to address unbounded network environments. This report describes the survivability approach to helping assure that a system that must operate in an unbounded network is robust in the presence of attack and will survive attacks that result in successful intrusions. Included are discussions of survivability as an integrated engineering framework, the current state of survivability practice, the specification of survivability requirements, strategies for achieving survivability, and techniques and processes for analyzing survivability.

Ellison, R. J. & Moore, A. P. Trustworthy Refinement Through Intrusion-Aware Design (CMU/SEI-2003-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
http://www.sei.cmu.edu/publications/documents/03.reports /03tr002.html.

High confidence in a system's survivability requires an accurate understanding of the system's threat environment and the impact of that environment on system operations. Unfortunately, existing development methods for secure and survivable information systems often employ a patchwork approach in which the focus is on deciding which popular security components to integrate rather than making a rational assessment of how to address the attacks that are likely to compromise the overall mission. This report proposes an intrusion-aware design model called trustworthy refinement through intrusion-aware design (TRIAD). TRIAD helps information system decision makers formulate and maintain a coherent, justifiable, and affordable survivability strategy that addresses mission-compromising threats for their organization. TRIAD also helps in evaluating and maintaining an information system design in terms of its ability to implement a survivability strategy. This report demonstrates the application of TRIAD to the refinement of a survivability strategy for a business that sells products over the Internet.

TRIAD provides a solid foundation for the further refinement, experimentation, and validation of an approach to exploit knowledge of intruder behavior to improve system architecture design and operations. Ultimately, with effective tool support and evidence of its efficacy, TRIAD will be integrated with more comprehensive life-cycle models for the development and maintenance of high-confidence systems.

Feather, M. S. “A Risk-Centric Decision Process.” Software Engineering for High Assurance Systems (SEHAS) 2003. Portland, OR, May 9-10, 2003. http://www.sei.cmu.edu/community/sehas-workshop/feather/.

Summary of Part I: Information: Make use of information available early in lifecycle. Combine knowledge from experts and past experience. Accommodate both evidence and estimates. Process: Gather the right information in the right way. Objectives, including their relative importance. Risks, and by how much they impact objectives and requirements. Mitigations, and by how much their use would reduce risk. Tool support: Effectively handle voluminous amounts of information. Capture experts’ knowledge on-the-fly during intensive sessions. Present information through cogent visualizations. Derive additional knowledge via calculation and search.

Part II includes an example of application of the risk-based reasoning to the “Technology infusion” problem. For illustration, model checking technology is considered.

Using the DDP risk-centric decision process, aspects application of model checking (for what, and by whom) are represented as DDP “Objectives”, impediments to these are represented as DDP “Risks”, and solutions to those impediments are represented as DDP “Migrations”. Quantitative (albeit coarse-grained) estimates of by how much each solution addresses each impediment allows for scrutiny of the net effects of decisions.

Firesmith, D. G., “Security Use Cases.” Journal of Object Technology 2, 3 (May-June 2003): 53-64. http://www.jot.fm/issues/issue_2003_05/column6.

Although use cases are a popular modeling approach for engineering functional requirements, they are often misused when it comes to engineering security requirements because requirements engineers unnecessarily specify security architectural mechanisms instead of security requirements. After discussing the relationships between misuse cases, security use cases, and security mechanisms, this column provides examples and guidelines for properly specifying essential (i.e., requirements-level) security use cases.

Firesmith, D. G. “Engineering Security Requirements.” Journal of Object Technology 2, 1 (January-February 2003): 53-68. http://www.jot.fm/issues/issue_2003_01/column6.

Most requirements engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them. They thus end up specifying architecture and design constraints rather than true security requirements. This article defines the different types of security requirements and provides associated examples and guildlines with the intent of enabling requirements engineers to adequately specify security requirements without unnecessarily constraining the security and architecture teams from using the most appropriate security mechanisms for the job.

Fu, Z.; Wu, S. F.; Huang, J.; Loh, K.; Gong, F.; Baldine, I.; & Xu, C. "IP-Sec/VPN Security Policy: Correctness, Conflict Detection and Resolution," 39-56. Policies for Distributed Systems and Networks. Edited by M. Sloman, J. Lobo, and E. C. Lupu. Proceedings of the International Workshop, POLICY 2001, Bristol, UK, Jan. 29-31, 2001. Berlin, Germany: Springer-Verlag, 2001 (ISBN 3540416102; Lecture Notes in Computer Science Vol. 1995).

IPSec (Internet Security Protocol Suite) functions will be executed correctly only if its policies are correctly specified and configured. Manual IP-Sec policy configuration is inefficient and error-prone. An erroneous policy could lead to communication blockade or serious security breach. In addition, even if policies are specified correctly in each domain, the diversified regional security policy enforcement can create significant problems for end-to-end communication because of interaction among policies in different domains. A policy management system is, therefore, demanded to systematically manage and verify various IPSec policies in order to ensure an end-to-end security service. This paper contributes to the development of an IPSec policy manage-ment system in two aspects. First, we defined a high-level security requirement, which not only is an essential component to automate the policy specification process of transforming from security requirements to specific IPSec policies but also can be used as criteria to detect conflicts among IPSec policies, i.e. policies are correct only if they satisfy all requirements. Second, we developed mechanisms to detect and resolve conflicts among IPSec policies in both intra-domain and inter-domain environment.

Gayash, A., Viswanathan, V., Padmanabhan, D., & Mead, N. R. SQUARE-Lite: Case Study on VADSoft Project, CMU/SEI-2008-SR-017. Software Engineering Institute, Carnegie Mellon University, 2008.
http://www.sei.cmu.edu/pub/documents/08.reports/08sr017.pdf

This report describes the SQUARE-Lite security requirements engineering method and the Security Quality Requirements Engineering (SQUARE) process, from which SQUARE-Lite is derived, and a student team presents the results of working with a client using SQUARE-Lite to develop security requirements for a financial application.

Gordon, D.; Mead, N. R.; Stehney, T.; Wattas, N.; & Yu, E. SQUARE Methodology: Case Study on Asset Management System, Phase II (CMU/SEI-2005-SR-005). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. CMU/SEI-2005-SR-005.

This report describes the second phase of an application of the System Quality Requirements Engineering (SQUARE) Methodology developed by the Software Engineering Institute's Networked Systems Survivability Program on an asset management system. An overview of the SQUARE process and the vendor is presented, followed by a description of the system under study. The research completed on Steps 4 through 9 of this nine-step process is then explained and feedback on its implementation is provided. The report concludes with a summary of findings and gives recommendations for future considerations of SQUARE testing.

This report is one of a series of reports resulting from research conducted by the SQUARE team as part of an independent research and development project of the Software Engineering Institute.

Haley, C. B., Laney, R., Moffett, J. D., & Nuseibeh, B. “Security Requirements Engineering: A Framework for Representation and Analysis.” IEEE Transactions on Software Engineering 34, 1 (Jan.-Feb. 2008): 133-153.

This paper presents a framework for security requirements elicitation and analysis. The framework is based on constructing a context for the system, representing security requirements as constraints, and developing satisfaction arguments for the security requirements. The system context is described using a problem-oriented notation, then is validated against the security requirements through construction of a satisfaction argument. The satisfaction argument consists of two parts: a formal argument that the system can meet its security requirements, and a structured informal argument supporting the assumptions expressed in the formal argument. The construction of the satisfaction argument may fail, revealing either that the security requirement cannot be satisfied in the context, or that the context does not contain sufficient information to develop the argument. In this case, designers and architects are asked to provide additional design information to resolve the problems. We evaluate the framework by applying it to a security requirements analysis within an air traffic control technology evaluation project.

Hatebur, D., Heisel, M., & Schmidt, H. "A Pattern System for Security Requirements Engineering," 356-365. Second International Conference on Availability, Reliability and Security (ARES'07). Vienna, Austria, 2007. IEEE Computer Society, 2007.

> We present a pattern system for security requirements engineering, consisting of security problem frames and concretized security problem frames. These are special kinds of problem frames that serve to structure, characterize, analyze, and finally solve software development problems in the area of software and system security. We equip each frame with formal preconditions and postconditions. The analysis of these conditions results in a pattern system that explicitly shows the dependencies between the different frames. Moreover, we indicate related frames, which are commonly used together with the considered frame. Hence, our approach helps security engineers to avoid omissions and to cover all security requirements that are relevant for a given problem.

Heitmeyer, C. & Bharadwaj, R. "Applying the SCR Requirements Method to the Light Control Case Study." Journal of Universal Computer Science 6, 7 (2000): 650-678.

> To date, the SCR (Software Cost Reduction) requirements method has been used in industrial environments to specify the requirements of many practical systems, including control systems for nuclear power plants and avionics systems. This paper describes the use of the SCR method to specify the requirements of the Light Control System (LCS), the subject of a case study at the Dagstuhl Seminar on Requirements Capture, Documentation, and Validation in June 1999. It introduces a systematic process for constructing the LCS requirements specification, presents the specification of the LCS in the SCR tabular notation, discusses the tools that we applied to the LCS specification, and concludes with a discussion of a number of issues that arose in developing the specification.

Heitmeyer, C.; Jeffords, R. D.; & Labaw, B.G. "Automated Consistency Checking of Requirements Specifications." ACM Transactions on Software Engineering and Methodology 5, 3 (April-June 1996): 231-261.

This article describes a formal analysis technique, called consistency checking, for automatic detection of errors, such as type errors, nondeterminism, missing cases, and circular definitions, in requirements specifications. The technique is designed to analyze requirements specifications expressed in the SCR (Software Cost Reduction) tabular notation. As background, the SCR approach to specifying requirements is reviewed. To provide a formal semantics for the SCR notation and a foundation for consistency checking, a formal requirements model is introduced; the model represents a software system as a finite-state automation, which produces externally visible outputs in response to changes in monitored environmental quantities. Results of two experiments are presented which evaluated the utility and scalability of our technique for consistency checking in a real-world avionics application. The role of consistency checking during the requirements phase of software development is discussed.

Helmer, G.; Wong, J.; Slagell, M.; Honavar, V.; Miller, L.; & Lutz, R. “A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System.” Requirements Engineering 7, 4 (December 2002): 207-220.

Requirements analysis for an intrusion detection system (IDS) involves deriving requirements for the IDS from analysis of the intrusion domain. When the IDS is, as here, a collection of mobile agents that detect, classify, and correlate system and network activities, the derived requirements include what activities the agent software should monitor, what intrusion characteristics the agents should correlate, where the IDS agents should be placed to feasibly detect the intrusions, and what countermeasures the software should initiate. This paper describes the use of software fault trees for requirements identification and analysis in an IDS. Intrusions are divided into seven stages (following Ruiu, 1999), and a fault subtree is developed to model each of the seven stages (reconnaissance, penetration, etc.). Two examples are provided. This approach was found to support requirements evolution (as new intrusions were identified), incremental development of the IDS, and prioritisation of countermeasures.

Heninger, K.; Parnas, D. L.; Shore, J. E.; & Kallander, J. W. “Software Requirements for the A-7E Aircraft.” Technical Report 3876. Washington, D.C.: Naval Research Laboratory, 1978.

The Software Cost Reduction (SCR) research project introduced a new approach to specifying requirements for real-time embedded systems. The principles were applied in the development of the Software Requirements of the A-7E Aircraft, as an example of the use of the approach. The system software requirements specification document comprises the first product in a series of products which the SCR methodology produces, The methodology is intended to be adaptable for various types of systems. Specification properties which it supports include: (1) conciseness, (2) preciseness, (3) aids to completeness, (4) avoidance of redundancy, (5) descriptions of all externally visible behavior, (6) ease of change, (7) good reference tool, (8) record of fundamental assumptions which might otherwise be only implicit, (9) record of responses to error conditions, (10) specification of constraints on the system, and (11) separation of concerns; that is, a division of the information into distinct, independent parts. Software Cost Reduction (SCR) project, Methodology, Software engineering, A-7E aircraft, Software requirement specification, Software development lifecycle.

Heninger, K. L. “Specifying Software Requirements for Complex Systems: New Techniques and their Application.” IEEE Transactions on Software Engineering SE-6, 1 (January 1980): 2-13.

This paper concerns new techniques for making requirements specifications precise, concise, unambiguous, and easy to check for completeness and consistency. The techniques are well suited for complex real-time software systems; they are developed to document the requirements of existing flight software for the Navy’s A-7 aircraft. The paper outlines the information that belongs to a requirements document and discusses the objectives behind the techniques. Each technique is described and illustrated with examples from the A-7 document. The purpose of the paper is to introduce the A-7 document as a model of a disciplined approach to requirements specification; the document is available to anyone who wishes to see a fully worked-out example of the approach.

Herrmann, D. & Keith, S. “Application of Common Criteria to Telecomm Services: A Case Study.” Computer Security Journal XVII, 2 (Spring 2001): 21-28.

The ISO/IEC 15408 standards, known as the Common Criteria, were developed to facilitate consistent evaluations of security products and systems. These standards represent an international effort to define an ITSEC evaluation methodology which would receive mutual recognition between customers and vendors throughout the global economy. The ISO/IEC 15408 standards were approved December 1999. Since then their primary application has been the certification of COTS products. The FAA has applied the Common Criteria to a large system; i.e., the development of a Protection Profile for the National Air Space (NAS) Infrastructure Management System (NIMS). The next step is to apply the Common Criteria methodology to a services contract. The FAA Telecommunications Infrastructure (FTI) project, discussed in the case study is one of the first projects to do so.

Hoglund, G. & McGraw, G. Exploiting Software. Boston, MA: Addison-Wesley, 2004.

Exploiting Software highlights the most critical part of the software quality problem. As it turns out, software quality problems are a major contributing factor to computer security problems. Increasingly, companies large and small depend on software to run their businesses every day. The current approach to software quality and security taken by software companies, system integrators, and internal development organizations is like driving a car on a rainy day with worn-out tires and no air bags. In both cases, the odds are that something bad is going to happen, and there is no protection for the occupant/owner. This book will help the reader understand how to make software quality part of the design.

Topics covered include how to make software fail, either by doing something it wasn't designed to do, or by denying its use to its rightful users. Techniques—including reverse engineering, buffer overflow, and particularly provision of unexpected input—are covered along with the tools needed to carry them out. A section on hardware viruses is detailed and frightening.

Hope, P.; Anton, A.; & McGraw, G. “Misuse and Abuse Cases: Getting Past the Positive.” IEEE Security & Privacy 2, 3 (May-June 2004): 32-34.

Software development is all about making software do something: when software vendors sell their products, they talk about what the products do to make customers' lives easier, such as encapsulating business processes or something similarly positive. Following this trend, most systems for designing software also tend to describe positive features. The authors provide a nonacademic introduction to the software security best practice of misuse and abuse cases, showing you how to put the basic science to work.

Ingalsbe, J. A., Kunimatsu, L., Baeten, T., & Mead, N. R. "Threat Modeling: Diving into the Deep End." IEEE Software 25, 1 (Jan./Feb. 2008): 28-34.

> Ford Motor Company is currently introducing threat modeling on strategically important IT applications and business processes. The objective is to support close collaboration between IT Security & Controls (the ITS group at Ford) and its business customers in analyzing threats and better understanding risk. To accomplish this, a core group of security personnel have piloted Microsoft’s Threat Analysis and Modeling process and tool on a dozen projects. Here, they discuss this TAM process, its benefits and challenges, and some deployment solutions.

Kienzle, D. M. & Wulf, W. A. “A Practical Approach to Security Assessment,” 5-16. Proceedings of the 1998 Workshop on New Security Paradigms. Charlottesville, VA, Sept. 22-26, 1998. New York, NY: ACM, 1998 (ISBN 0897919866).

> Conventional approaches to building and assessing security critical software are based on the implicit assumption that security is the single most important concern and can be the primary factor driving the software development process. Changes in the marketplace and the nature of security requirements have brought this assumption into question. There is now a large class of systems in which security must compete with other development goals. A risk-driven process model of software development provides a framework for building software that balances conflicting requirements, but a risk-driven process invalidates many of the assumptions made by conventional approaches to the specification and verification of security requirements. This paper presents a new approach to assessing the degree to which software meets its security requirements. It does not propose a new specification notation or analysis technique, but provides a general framework into which existing notations and techniques can be integrated. It allows varying degrees of formality to be used: both across the components of the system and through the development process. The appropriate degree of formality is whatever degree proves necessary to satisfy the stakeholders in the system that the security goals have been met. This approach has been found to be theoretically appealing as well as useful in practice. We give a brief overview of the approach, explain how it integrates into a risk-driven process model, and discuss our early results in using it to assess, and thereby thus guide the development of, the Legion security model.

Knorr, K. & Rohrig, S. “Security Requirements of E-Business Processes,” 73-86. Towards the E-Society: E-Commerce, E-Business, and E-Government. Edited by B. Schmid, K. Stanoevska-Slabeva, and V. Tschammer. First IFIP Conference on E-Commerce, E-Business, E-Government, Zurich, Switzerland, Oct. 4-5, 2001. Norwell, MA: Kluwer Academic Publishers, 2001 (ISBN 0792375297).

The paper presents an open framework for the analysis of security requirements of business processes in electronic commerce. The most important dimensions of the framework are security objectives (confidentiality, integrity, availability, accountability), the phases of and the places/parties involved in the process. The approach is of an open nature so that it can be adapted to the heterogeneous needs of different application scenarios. The discussion of business processes within a virtual shopping mail illustrates the capacity and potential of the framework.

Labuschagne, L. “A Framework for Electronic Commerce, Security,” 441-50. Information Security for Global Information Infrastructures. Edited by S. Qing and J.H.P. Eloff. Fifteenth Annual Working Conference on Information Security, Beijing, China, Aug. 22-24, 2000. Norwell, MA: Kluwer Academic Publishers, 2000 (ISBN 0792379144).

The paper suggests a framework that can be used to identify the security requirements for a specific electronic commerce environment. The first step is to list all the security requirements for an electronic commerce environment in general. Next, all participants need to be identified. This is followed by the breaking down of the transactions into different autonomous actions. These actions are then mapped onto the participants involved, which serve as a model for the electronic commerce environment. This information is then used to identify the security requirements for a secure electronic commerce environment. The security requirements, in turn, are then used to develop the security architecture, consisting of appropriate security procedures and mechanisms and policy.

Leitold, H. & Posch, R. “ATM Network Security: Requirements, Approaches, Standards, and the SCAN Solution,” 191-204. Intelligence in Networks, SMARTNET ’99. Pathumthani, Thailand, Nov. 22-26, 1999. Boston, MA: Kluwer Academic Publishers, 1999.

Broadband networks based on the asynchronous transfer mode (ATM) are emerging rapidly. Both the technological component in terms of ATM infrastructure, as well as the area of applications requiring quality of service (QoS) by the means of bandwidth or delay constraints are covered by a variety of projects and products. However, given the increasing interest in applications such as governmental communication, transmission of medical information, or commercial applications, the necessity of providing secure means of delivering sensitive contents is apparent. We focus on security services in ATM networks. The variety of different approaches and solutions are categorised by the means of common and distinct functionality, as well as certain advantages and disadvantages. In addition, the standardisation efforts by the leading group in that area—the ATM Forum—are outlined. Finally, the essentials of the project SCAN are given, resulting in a comprehensive solution to security services in ATM networks.

Leiwo, J. "A Mechanism for Deriving Specifications of Security Functions in the CC Framework," 416-425. 10th International Workshop on Database and Expert Systems Applications. Florence, Italy, Sept. 1-3, 1999. Berlin, Germany: Springer-Verlag, 1999.

At the first stage of the common criteria process for evaluating the security of information systems, organizational objectives for information security are translated into the specification of all relevant security functions of a becoming system. These specifications are then assessed to specify the subset to be implemented, and further evaluated. The second stage involves risk analysis or related technologies, and the evaluation phase is the major contribution of the common criteria. The derivation of security function specifications from security objectives is the area where further research is needed to provide pragmatic tools for supporting the task. This paper describes a mechanism, harmonization of information security requirements, that aids in this process.

Leiwo, J.; Gamage, C.; & Zheng, Y. "Organizational Modeling for Efficient Specification of Information Security Requirements," 247-260. Advances in Databases and Information Systems: Third East European Conference, ADBIS'99. Maribor, Slovenia, Sept. 13-16, 1999. Berlin, Germany: Springer-Verlag, 1999 (Lecture Notes in Computer Science Vol. 1691).

Functional security requirements of information systems can roughly be classified into two: computer security requirements and communications security requirements. Challenges for developing notations for expressing these requirements are numerous, most importantly the difficulty of dealing with layers of ion, flexibility to adapt into many types of requirements, groupings of requirements, and requirement dependencies. Many frameworks for dealing with information security highlight the importance of a properly defined organization of security but fail to establish models to support the specification. This paper establishes one such model and demonstrates how the above difficulties can be overcome through extensive application of organizational modeling of information security.

Leveson, N. G. Safeware: System Safety and Computers. Reading, MA: Addison-Wesley, 1995.

Summary Software engineers and system developers need to understand the issues and develop the skills required to prevent destructive accidents before they occur. This book examines what is currently known about building safe electromechanical systems and looks at past accidents to see what lessons can be applied to new computer-controlled systems.

Linger, R. C.; Mead, N. R.; & Lipson, H. F. “Requirements Definition for Survivable Systems,” 14-23. Third International Conference on Requirements Engineering. Colorado Springs, CO, April 6-10, 1998. Los Alamitos, CA: IEEE Computer Society, 1998.

Pervasive societal dependency on large scale, unbounded network systems, the substantial risks of such dependency, and the growing sophistication of system intruders, have focused increased attention on how to ensure network system survivability. Survivability is the capacity of a system to provide essential services even after successful intrusion and compromise, and to recover full services in a timely manner. Requirements for survivable systems must include definitions of essential and non essential services, plus definitions of new survivability services for intrusion resistance, recognition, and recovery. Survivable system requirements must also specify both legitimate and intruder usage scenarios, and survivability practices for system development, operation, and evolution. The paper defines a framework for survivable systems requirements definition and discusses requirements for several emerging survivability strategies. Survivability must be designed into network systems, beginning with effective survivability requirements analysis and definition.

Linger, R. C.; Walton, G.; Pleszkoch, M. G.; & Hevner, A. R. “Flow-Service-Quality (FSQ) Requirements Engineering for High Assurance Systems.” http://www.cert.org/archive/pdf/FSQengineeringRHAS-02paper.pdf.

> Modern society could hardly function without the large-scale, network-centric information systems that pervade government, defense, and industry. These systems exhibit shifting boundaries, often unknown component behavior, constantly varying function and usage, and pervasive asynchronous operations. Their complexity challenges intellectual control and their survivability has become an urgent priority. Requirements engineering methods based on solid theoretical foundations and the realities of network systems are required to manage complexity and ensure survivability. Flow-Service Quality (FSQ) engineering is an emerging technology for requirements specification, development and evolution of network-centric systems. FSQ engineering is based of Flow Structures, Computational Quality Attributes, and Flow Management Architectures. These technologies can provide stable engineering foundations for the dynamic and unpredictable world of large-scale network systems. Flow structure requirements define mission task flows and their requirements into uses of system services in network traversals. Flows are deterministic for human understanding, despite the asynchronism of network operations. They can be defined, abstracted and verified with precision, and deal explicitly with uncertain factors, including uncertain COTS functionality and system failures and compromises. Computational quality attributes go beyond static, a priori estimates to define requirements for quality attributes such as reliability and survivability as dynamic functions to be computed in system operation.

Linger, R. C. Applying FSQ Engineering Foundations to Automated Calculation of Program Behavior (CMU/SEI-2003-TN-003). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.cert.org/archive/pdf/03tn003.pdf.

No software engineer can say with assurance how a sizable program, with its virtually infinite number of possible execution paths, will behave, that is, what it will do, in all circumstances of use. This incredible reality, widely acknowledged but little discussed, lies at the heart of intractable problems experienced in software development and use over the past 40 years. If full behavior is unknown, so too are embedded errors, vulnerabilities, and malicious code that can emerge in use. While this reality has seemed inevitable in the past, it need not be so in the future. The SEI CERT Research Center has been conducting research on Flow-Service-Quality (FSQ) engineering for complex, network-centric system analysis and development. FSQ Flow Structures treat the control structures of programs as rules, or implementations, of mathematical functions, that is, mappings from domains to ranges. The function, or behavior, of any control structure can be abstracted into a procedure-free statement that specifies its net functional effect in all circumstances of use with mathematical precision. The finite number of control structures in a program can be abstracted in stepwise fashion in an algebra of functions, to arrive at a precise statement of the program's overall behavior. The mathematical foundations largely exist, and development of such a capability is feasible, albeit difficult. Automated program behavior calculation would have a dramatic effect on software and systems engineering, and enable a new level of assurance in trustworthy systems. This report briefly summarizes research to date on Flow Structures and describes the application of their function-theoretic mathematical foundations to the problem of program behavior calculation.

Massacci, F., Mylopoulos, J., & Zannone, N. “Computer-aided Support for Secure Tropos.” Automated Software Engineering 14 (August 2007): 341-364.

In earlier work, we have introduced Secure Tropos, a requirements engineering methodology that extends the Tropos methodology and is intended for the design and analysis of security requirements. This paper briefly recaps the concepts proposed for capturing security aspects, and presents an implemented graphical CASE tool that supports the Secure Tropos methodology. Specifically, the tool supports the creation of Secure Tropos models, their translation to formal specifications, as well as the analysis of these specifications to ensure that they comply with specific security properties. Apart from presenting the tool, the paper also presents a two-tier evaluation consisting of two case studies and an experimental evaluation of the tool’s scalability.

McDermott, J. “Abuse-Case-Based Assurance Arguments,” 366-374. Proceedings 17th Annual Computer Security Applications Conference. New Orleans, LA, Dec. 10-14, 2001. Los Alamitos, CA: IEEE Computer Society Press, 2001.

This paper describes an extension to abuse-case-based security requirements analysis that provides a lightweight means of increasing assurance in security relevant software. The approach is adaptable to lightweight software development processes but results in a concrete and explicit assurance argument. Like abuse-case-based security requirements analysis, this approach is suitable for use in projects without security experts. When used in this way (without security experts) it will not produce as much assurance as the more traditional alternatives, but arguably give better results than ad hoc consideration of security issues.

McDermott, J., Fox, C., “Using Abuse Case Models for Security Requirements Analysis,” 55-64. Proceedings 15th Annual Computer Security Applications Conference. Scottsdale, AZ, Dec. 6-10, 1999. Los Alamitos, CA: IEEE Computer Society Press, 1999.

The relationships between the work products of a security engineering process can be hard to understand, even for persons with a strong technical background but little knowledge of security engineering. Market forces are driving software practitioners who are not security specialists to develop software that requires security features. When these practitioners develop software solutions without appropriate security-specific processes and models, they sometimes fail to produce effective solutions. We have adapted a proven object oriented modeling technique, use cases, to capture and analyze security requirements in a simple way. We call the adaptation an abuse case model. Its relationship to other security engineering work products is relatively simple, from a user perspective.

Mead, N. R.; Linger, R. C.; McHugh, J.; & Lipson, H. F. “Managing Software Development for Survivable Systems.” Annals of Software Engineering 11 (2001): 45-78.

The environment in which software projects are managed has evolved dramatically in recent years. This evolution has been driven by an extraordinary increase in network connectivity and extensive use of contractors for system development, raising issues of interoperability, security, ownership, and intellectual property rights. Project managers face the ongoing challenge of creating an orderly incremental development process, which often proceeds for years, in this complex environment. At the same time, the dependency of organizations, their suppliers, and their customers on complex, large-scale information systems is increasing at an astonishing rate, to the point that conduct of business operations is virtually impossible if these systems are compromised. As a result, survivability is receiving, increasing attention as a key property of critical systems. Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. Given the severe consequences of system failure, it is clear that many more organizations should be, and at present are not, concerned with survivability issues. However, when survivability is added to the project management equation, software life cycles can look rather different from the traditional life-cycle model. In this paper we discuss this changing software project management environment, the impact of system survivability, and life-cycle activities that are tailored to development and evolution of survivable systems. Achieving survivable systems requires that survivability be integrated into project life cycles, and not treated as an add-on property.

Mead, N. R. “Survivability Requirements: How Can We Assess Them Versus Other Requirements for High Assurance Systems,” 65-68. International Workshop on Requirements for High Assurance Systems, Essen, Germany, Sept. 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

High assurance systems (HASs) are computer systems where compelling evidence is required that the system delivers its services in a manner that satisfies certain critical properties. Among the critical properties are security properties (i.e., the system prevents unauthorized disclosure, modification and withholding of sensitive information), safety properties (the system prevents unintended events that could result in death, injury, illness, or property damage), survivability properties (the system continues to fulfill its mission in the presence of attacks, accidents, or failures), fault-tolerant properties (the system guarantees a certain quality of service despite faults, such as hardware, workload, or environmental anomalies), and real-time properties (the system delivers its outputs within specified time intervals).

Mead, N. R. International Liability Issues for Software Quality (CMU/SEI-2003-SR-001, ADA416434). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03sr001.html.

This report focuses on international law related to cybercrime, international information security standards, and software liability issues as they relate to information security for critical infrastructure applications. Each area is explored and implications for U.S. policy and efforts to create cyber security policy worldwide are discussed. Recommendations are made for U.S. government participation and leadership.

This report is one of a series of reports on U.S. policy by the CERT Coordination Center[R]. Prior reports focused on international infrastructure for global security incident response and the technical challenges and global policy issues of tracking and tracing cyber attacks.

Mead, N. R. Requirements Engineering for Survivable Systems (CMU/SEI-2003-TN-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tn013.html.

This report describes the current state of requirements engineering for survivable systems, that is, systems that are able to complete their mission in a timely manner, even if significant portions are compromised by attack or accident. Requirements engineering is defined and requirements engineering activities are described. Survivability requirements are then explained, and requirements engineering methods that may be suitable for survivable systems are introduced. The report concludes with a summary and a plan for future research opportunities in survivable systems requirements engineering.

Mead, N. R. "Requirements Elicitation and Analysis Processes for Safety & Security Requirements." Fourth International Workshop on Requirements for High Assurance Systems, September 6, 2004, Kyoto, Japan. Pittsburgh, PA: Software Engineering Institute, 2004. http://www.sei.cmu.edu/community/rhas-workshop/#papers.

This paper describes a process framework for the elicitation of safety and security requirements and early experience in applying the framework. A larger research project that provides context for the process is briefly described.

Mead, N. R. "Experiences in Eliciting Security Requirements." CrossTalk 19, 12 (December 2006): 14-19.

There are many requirements elicitation methods, but we seldom see elicitation performed specifically for security requirements. One reason for this is that few elicitation methods are specifically directed at security requirements. Another factor is that organizations seldom address security requirements elicitation specifically and instead lump them in with other traditional requirements elicitation methods. This article describes an approach for doing trade-off analysis among requirements elicitation methods. Several case studies were conducted in security requirements elicitation; the detailed results of one case study and brief results of two other case studies are presented here.

Mead, N. R. “Identifying Security Requirements Using the Security Quality Requirements Engineering (SQUARE) Method,” in Integrating Security and Software Engineering: Advances and Future Visions, H. Mouratidis and P. Giorgini, Eds.: Idea Group, 2006, pp. 44-69.

In this chapter we describe general issues in developing security requirements, methods that have been useful, and a process (SQUARE) that can be used for eliciting, analyzing, and documenting security requirements for software systems. SQUARE, which was developed by the CERT program at Carnegie Mellon University’s Software Engineering Institute, provides a systematic approach to security requirements engineering. SQUARE has been used on a number of client projects by Carnegie Mellon student teams, prototype tools have been developed, and research is ongoing to improve this promising method.

Mead, N. R. How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods (CMU/SEI-2007-TN-021). Software Engineering Institute, Carnegie Mellon University, 2007.
http://www.sei.cmu.edu/pub/documents/07.reports/07tn021.pdf

The Security Quality Requirements Engineering (SQUARE) method, developed at the Carnegie Mellon Software Engineering Institute, provides a systematic way to identify security requirements in a software development project. This report describes SQUARE and then describes other methods used for identifying security requirements, such as the Comprehensive, Lightweight Application Security Process, the Security Requirements Engineering Process, and Tropos, and compares them with SQUARE. The report concludes with some guidelines for selecting a method and a look at some related trends in requirements engineering.

Mead, N. R. "Identifying Security Requirements Using the Security Quality Requirements Engineering (SQUARE) Method," in Information Security and Ethics: Concepts, Methodologies, Tools, and Applications, H. Nemati, Ed.: IGI Global, 2007, pp. 943-963.

> In this chapter we describe general issues in developing security requirements, methods that have been useful, and a process (SQUARE) that can be used for eliciting, analyzing, and documenting security requirements for software systems. The SQUARE process, developed at SEI/CERT, provides a systematic approach to security requirements engineering. The method has been used on a number of client projects by CMU student teams, prototype tools have been developed, and research is ongoing to improve this promising method.

Mead, N. R., & Hough, E. "Security Requirements Engineering for Software Systems: Case Studies in Support of Software Engineering Education," 149-156. Conference on Software Engineering Education and Training. Turtle Bay, Hawaii, April 2006. IEEE Computer Society.

> Software engineering curricula too often neglect the development of security requirements for software systems. As a consequence, programmers often produce buggy code with weak security measures. This report focuses on three case studies in which graduate students applied a novel security requirements engineering methodology to real-world software development projects. The experiences showed promise for curriculum integration in educating students about the importance of security requirements in software engineering, as well as how to develop such requirements. © 2006 IEEE.

Mead, N. R., & Shoemaker, D. "Novel Methods of Incorporating Security Requirements Engineering into Software Engineering Courses and Curricula," in Software Engineering: Effective Teaching and Learning Approaches and Practices, D. Ellis, & Naveda, Ed.: IGI Global, 2008, pp. 98-113.

This chapter describes methods of incorporating security requirements engineering into software engineering courses and curricula. The chapter discusses the importance of security requirements engineering and the relationship of security knowledge to general computing knowledge by comparing a security body of knowledge to standard computing curricula. Then security requirements is related to standard computing curricula and educational initiatives in security requirements engineering are described, with their results. An expanded discussion of the SQUARE method in security requirements engineering case studies is included, as well as future plans in the area. Future plans include the development and teaching of academic course materials in security requirements engineering, which will then be made available to educators. The authors hope that more educators will be motivated to teach security requirements engineering in their software engineering courses and to incorporate it in their curricula.

Mead, N. R. & Stehney, T. "Security Quality Requirements Engineering (SQUARE) Methodology." Software Engineering for Secure Systems (SESS05), ICSE 2005 International Workshop on Requirements for High Assurance Systems, St. Louis, MO, May 15-16, 2005. http://homes.dico.unimi.it/%7Emonga/sess05.html.

Requirements engineering, a vital component in successful project development, often neglects sufficient attention to security concerns. Further, industry lacks a useful model for incorporating security requirements into project development. Studies show that upfront attention to security saves the economy billions of dollars. Industry is thus in need of a model to examine security and quality requirements in the development stages of the production lifecycle.In this paper, we examine a methodology for both eliciting and prioritizing security requirements on a development project within an organization. We present a model developed by the Software Engineering Institute's Networked Systems Survivability (NSS) Program, and then examine two case studies where the model was applied to a client system. The NSS Program continues to develop this useful model, which has proven effective in helping an organization understand its security posture.

Mead, N. R.; Hough, E.; & Stehney, T. Security Quality Requirements Engineering (SQUARE) Methodology (CMU/SEI-2005-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html.

Requirements engineering, a vital component in successful project development, often does not include sufficient attention to security concerns. Studies show that up-front attention to security can save the economy billions of dollars, yet security concerns are often treated as an afterthought to functional requirements. Industry can thus benefit from a model to examine security requirements in the development stages of the production life cycle.

This report presents the Security Quality Requirements (SQUARE) Methodology for eliciting and prioritizing security requirements in software development projects, which was developed by the Software Engineering Institute's Networked Systems Survivability (NSS) Program. The methodology's steps are explained, and results from its application in recent case studies are examined. The NSS Program continues to develop SQUARE, which has proven effective in helping organizations understand their security posture and produce products with verifiable security requirements.

Mead, N. R., Viswanathan, V., & Padmanabhan, D. “Incorporating Security Requirements Engineering into the Dynamic Systems Development Method,” 949-954. International Workshop on Security and Software Engineering at COMPSAC 2008. Turku, Finland, 2008. IEEE Computer Society, 2008.

This paper provides a roadmap for addressing security requirements on projects using an agile approach. The Dynamic Systems Development Method (DSDM) is used as an example framework for development. Security Quality Requirements Engineering (SQUARE) is used as an example methodology to address security issues early in the development life cycle. SQUARE can be more effective when it fits into an organization's existing development process. Hence this paper describes a way to fit the SQUARE methodology into the DSDM framework. © 2008 IEEE.

Mead, N. R., Viswanathan, V., Padmanabhan, D., & Raveendran, A. Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models (CMU/SEI-2008-TN-006, ADA482308). Carnegie Mellon University, Software Engineering Institute, 2008.
http://www.sei.cmu.edu/pub/documents/08.reports/08tn006.pdf

SQUARE (Security Quality Requirements Engineering) is a method for eliciting and prioritizing security requirements in software development projects. This report describes how SQUARE can be incorporated in standard life-cycle models for security-critical projects. Life-cycle models and process methods considered for the report are the waterfall model, Rational Unified Process, the spiral model, and Dynamic Systems Development Method (an agile method).

This report is for information technology managers and security professionals, management personnel with technical and information security knowledge, and any personnel who manage security-critical projects that follow standard life-cycle models.

Mead, N. R., Viswanathan, V., & Zhan, J. "Incorporating Security Requirements Engineering into the Rational Unified Process," 537-542. International Conference on Information Security and Assurance (ISA). Busan, Korea, 2008. IEEE Computer Society, 2008.

This paper provides a roadmap for developing security-critical projects using rational unified process as a framework for development. The security quality requirements engineering (SQUARE) methodology provides a way to address security issues early in the development lifecycle. SQUARE can be more effective when it fits into an organization's existing development process. Hence this paper describes a way to fit the SQUARE methodology into the rational unified process.

Mellado, D., Fernández-Medina, E., & Piattini, M. "A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems." Computer Standards & Interfaces 29, 2 (February 2007): 244-253. http://www.sciencedirect.com

In order to develop security critical Information Systems, specifying security quality requirements is vitally important, although it is a very difficult task. Fortunately, there are several security standards, like the Common Criteria (ISO/IEC 15408), which help us handle security requirements. This article will present a Common Criteria centred and reuse-based process that deals with security requirements at the early stages of software development in a systematic and intuitive way, by providing a security resources repository as well as integrating the Common Criteria into the software lifecycle, so that it unifies the concepts of requirements engineering and security engineering. © 2006 Elsevier B.V. All rights reserved.

Mellado, D., Fernández-Medina, E., & Piattini, M. "Applying a Security Requirements Engineering Process," 192-206. 11th European Symposium on Research in Computer Security. Hamburg, Germany: Springer, 2006 (LNCS 4189).

Nowadays, security solutions are mainly focused on providing security defences, instead of solving one of the main reasons for security problems that refers to an appropriate Information Systems (IS) design. In fact, requirements engineering often neglects enough attention to security concerns. In this paper it will be presented a case study of our proposal, called SREP (Security Requirements Engineering Process), which is a standard-centered process and a reuse-based approach which deals with the security requirements at the earlier stages of software development in a systematic and intuitive way by providing a security resources repository and by integrating the Common Criteria into the software development lifecycle. In brief, a case study is shown in this paper demonstrating how the security requirements for a security critical IS can be obtained in a guided and systematic way by applying SREP.

Mellado, D., Fernández-Medina, E., & Piattini, M. "Secure Information Systems Development," 467-470. Proceedings of the International Conference on Security and Cryptography. Setúbal, Portugal, August 7-10, 2006. Institute for Systems and Technologies of Information, Control and Communication, 2006 (ISBN: 972-8865-63-5).

Integration of security into the early stages of the system development is necessary to build secure systems. However, in the majority of software projects security is dealt with when the system has already been designed and put into operation. This paper will propose an approach called SREP (Security Requirements Engineering Process) for the development of secure software. We will present an iterative and incremental micro-process for the security requirements analysis that is repeatedly performed at each phase. It integrates the Common Criteria into the software lifecycle model as well as it is based on the reuse of security requirements, by providing a security resources repository. In brief, we will present an approach which deals with the security requirements at the early stages of software development in a systematic and intuitive way, and which also conforms to ISO/IEC 17799:2005.

Mellado, D., Fernández-Medina, E., & Piattini, M. "Security Requirements Variability for Software Product Lines," 1413-1420. The Third International Conference on Availability, Reliability and Security. Barcelona, Spain, March 2008. IEEE Computer Society, 2008.

Software product line engineering has proven to be one of the most successful paradigms for developing a diversity of similar software applications and software-intensive systems at low costs, in short time, and with high quality, by exploiting commonalities and variabilities among products to achieve high levels of reuse. At the same time, due to the complexity and extensive nature of product line development, security and requirements engineering are critical success factors in the development of a software product line. However, most of the current product line practices in requirements engineering do not adequately address the security requirements engineering. Therefore, in this paper we will propose a security requirements decision model driven by security standards along with a security variability model to manage the variability of the security requirements related artefacts. The aim of this approach is to deal with security requirements from the early stages of the product line development in a systematic way, in order to facilitate the conformance to the most relevant security standards with regard to the management of security requirements, such as ISO/IEC 27001 and ISO/IEC 15408.

Mendiratta, V. B. "Assessing the Reliability Impacts of Software Fault-Tolerance Mechanisms," 99-103. Proceedings of the 7th International Symposium on Software Reliability Engineering. White Plains, NY, Oct. 30-Nov. 2, 1996. New York, NY: IEEE Computer Society Press, 1996.

Telecommunications systems are characterized by highly stringent reliability requirements for system availability and defect rate. A combination of approaches is used to achieve high software reliability, namely, fault avoidance, fault removal and implementation of fault-tolerant mechanisms. This paper focuses on the implementation of software fault-tolerant mechanisms and analyzes the impact of these mechanisms on software reliability. Based on field data on the frequency of invocation of some fault-tolerant mechanisms, we present an escalating recovery model for predicting the impact of these mechanisms on lost calls. The key parameters of the model are: the software fault recovery coverage factor; the proportion of successful recoveries at each level and the calls lost at each recovery level. The output of the model is a distribution and average of the number of lost calls per software error. The applicability of this model to systems with high reliability has been validated; the applicability of the model to less reliable systems is an area for future work.

Mills, H. D. "Certifying the Correctness of Software," 373-381. Proceedings of the 25th Hawaii International Conference on System Sciences, Vol. 2. Kauai, Hawaii, Jan. 7-10, 1992. Los Alamitos, CA: IEEE Computer Society Press, 1992.

Software is either correct or incorrect in design to a specification in contrast to hardware which is reliable to a certain level to a correct design. Software of any size or complexity can only be tested partially, and typically a very small fraction of possible inputs are actually tested. Certifying the correctness of such software requires two conditions, namely (1) statistical testing with inputs characteristic of actual usage, and (2) no failures in the testing. If any failures arise in testing or subsequent usage, the software is incorrect, and the certification invalid. If such failures are corrected, the certification process can be restarted, with no use of previous testing.

Moffett, Jonathan D.; Haley, Charles B.; & Nuseibeh, Bashar. Core Security Requirements Artefacts (Technical Report 2004/23). Milton Keynes, UK: Department of Computing, The Open University, June 2004.

Although security requirements engineering has recently attracted increasing attention, it has lacked a context in which to operate. A number of papers have described how security requirements may be violated, but apart from a few hints in the general literature, none have described satisfactorily what security requirements are.

This paper proposes a framework of core security requirements artefacts, which unifies the concepts of the two disciplines of requirements engineering and security engineering. From requirements engineering it takes the concept of functional goals, which are operationalised into functional requirements, with appropriate constraints. From security engineering it takes the concept of assets, together with threats of harm to those assets. Security goals aim to protect from those threats, and are operationalised into security requirements, which take the form of constraints on the functional requirements.

In addition we explore the consequences of the fact that security is concerned with the protection of assets, while computers only provide interfaces. We show how to specify the relationship between security requirements and the specification of software behaviour, using Jackson's Problem Frames approach.

Moore, A. P.; Ellison, R. J.; & Linger, R. C. Attack Modeling for Information Security and Survivability (CMU/SEI-2001-TN-001, ADA388771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.sei.cmu.edu/publications/documents/01.reports/01tn001.html.

> Many engineering disciplines rely on engineering failure data to improve their designs. Unfortunately, this is not the case with information system engineers, who generally do not use security failure data¾particularly attack data--to improve the security and survivability of systems that they develop. Part of the reason for this is that, historically, businesses and governments have been reticent to disclose information about attacks on their systems for fear of losing public confidence or for fear that other attackers would exploit the same or similar vulnerabilities. Specific, detailed attack data has just not been available.
>
> However, increased public interest and media coverage of the Internet's security have resulted in increased publication of attack data in books, Internet newsgroups, and CERT security advisories, for example. Engineers can now use this data in a structured way to improve information system security and survivability.
>
> This technical note describes and illustrates an approach for documenting attack information in a structured and reusable form. We expect that security analysts can use this approach to document and identify commonly occurring attack patterns, and that information system designers and analysts can use these patterns to develop more survivable information systems.

Musa, J. D.; Iannino, A.; & Okumoto, K. Software Reliability: Measurement, Prediction, and Application. New York, NY: McGraw-Hill, 1987.

> This is one of the only books completely dedicated to practical applications of software reliability, measurement and evaluation. The book follows a natural and logical progression: Part I provides an introductory overview of the field. Part II proceeds to give a practical guide for applying software reliability measurement in such areas as system engineering, software project monitoring, scheduling and planning, software change management, and evaluation of software engineering technology. Concepts are explained through the use of graphs, examples, and case studies taken from a variety of sources, including actual software development projects. No previous background in statistics is necessary.

National Institute of Standards and Technology. “Software Errors Cost U.S. Economy $59.5 Billion Annually” (NIST 2002-10). http://www.nist.gov/public_affairs/releases/n02-10.htm (2002).

The estimate comes from a newly released study commissioned by the Department of Commerce's National Institute of Standards and Technology (NIST). At the national level, over half of the costs are borne by software users and the remainder by software developers/vendors. The study also found that, although all errors cannot be removed, more than a third of these costs could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal. Within the CAD/CAM/CAE or PDM category of software, approximately 60 percent of the automotive and aerospace manufacturers surveyed reported significant software errors in the previous year. Respondents who experienced errors reported an average of 40 major and 70 minor software bugs per year in their CAD/CAM/CAE or PDM software systems. The total cost impact on these manufacturing sectors from an inadequate software-testing infrastructure is estimated to be $1.8 billion, and the potential cost reduction from feasible infrastructure improvements is $600 million. Users of CAD/CAM/CAE and PDM software absorb approximately 75 percent of the total impact, with the automotive industry representing about 65 percent and the aerospace industry representing 10 percent. Software developers experience the remaining 25 percent of the costs.

Ortalo, R. “A Flexible Method for Information System Security Policy Specification,” 67-84. 5th European Symposium on Research in Computer Security – Proceedings. Louvain-la-Neuve, Belgium, Sept. 16-18. Berlin, Germany: Springer-Verlag, 1998. (Lecture Notes in Computer Science Vol. 1485.)

This paper presents a method for the specification of the security of information systems. The proposed approach provides a flexible and expressive specification method, corresponding to the specific needs of organizations. First, the paper outlines the overall guidelines of the security policy definition process, and the different consistency issues associated to the description of the security requirements of an organization information system. The specification language used is based on a convenient extension of deontic logic. The formalism and its extensions are defined briefly. To illustrate the use of this formalism, the paper presents how the method applies to the description of the security requirements of a real organization: a medium-size bank agency.

Paulk, M. C.; Weber, C. V.; Curtis, B.; & Chrissis, M. B. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, Reading, MA, 1994.

This book provides a description and technical overview of the Capability Maturity Model® (CMM®) framework, with guidelines for improving software process management. The CMM framework provides software professionals in government and industry with the ability to identify, adopt, and use sound management and technical practices for delivering software on time and within budget.

Protection Profile (PP) Consistency Guidance for Basic Robustness, Release 1.1, September 2002. http://www.iatf.net/protection_profiles/file_serve.cfm?chapter=guidance.pdf.

A Protection Profile (PP) is an implementation-independent specification of information assurance security requirements. Protection profiles are a complete combination of security objectives, security related functional requirements, information assurance requirements, assumptions, and rationale.

The purpose of a PP is to state a security problem rigorously for a given collection of system or products—known as the Target of Evaluation (TOE)—and to specify security requirements to address that problem without dictating how these requirements will be implemented.

Product vendors may respond to the security concerns defined by a PP by producing a Security Target (ST), which is similar to a PP except that it contains implementation-specific information that demonstrate how their product addresses those security concerns.

In accordance with their respective responsibilities under Public Law 100-235 (Computer Security Act of 1987), the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) have agreed to cooperate on the development of security requirements for key technology areas necessary for the protection of Federal information systems and networks, including those comprising the critical infrastructure within the United States. NIST and NSA are undertaking this effort:

- To ensure the U.S. Government has a consistent comprehensive set of recommended protection profiles for key technology areas;
- To forge partnerships with public and private sector constituencies to develop and gain consensus on PPs important for critical infrastructure protection; and
- To facilitate national and international convergence of protection profiles in key technology areas.

Purdue University. Second Symposium on Requirements Engineering for Information Security, Raleigh, NC, October 16, 2002. Lafayette, IN: CERIAS, Purdue University, 2002.

Security requirements for new eCommerce and Internet applications exceed the traditional requirements for network security and traditional software systems. Security requirements are more complex and increasingly critical. Informally stated and de facto requirements are often of critical importance in the design and operation of these systems, but are frequently not taken into account. The second symposium on requirements engineering for information security invited theoretical, experimental, and experience papers on a diversity of topics, particularly ones that pointed out new directions.

SREIS provided researchers and practitioners from various disciplines with a highly interactive forum to discuss security and privacy-related requirements. Specifically, attendance was encouraged from those in the fields of requirements engineering, software engineering, information systems, information and network security and trusted systems as well as those interested in approaches to analyzing, specifying, and testing requirements to increase the level of security provided to users interacting with pervasive commerce, research and government systems.

Reiser, H. & Vogt, G. "Security Requirements for Management Systems using Mobile Agents," 160-165. Proceedings ISCC2000—Fifth IEEE Symposium on Computers and Communications. Edited by S. Tohme and M. Ulema. Antibes, France, July 4-7, 2000. Los Alamitos, CA: IEEE Computer Society, 2000 (ISBN 0769507220).

Flexible and distributed management systems based on mobile agents have certain advantages over centralized and static management architectures. However, security plays a decisive role in terms of acceptance and applicability of mobile agents. In this paper we analyze the threats and attacks against mobile agent systems used for management purposes. Therefore, general models of mobile agent based management systems are developed. Based on a risk analysis of these models we derive security requirements. In order to satisfy these requirements components and services are identified and integrated in a comprehensive security architecture.

Rosado, D. G., Gutiérrez, C., Fernández-Medina, E., & Piattini, M. "Security Patterns and Requirements for Internet-Based Applications." Internet Research 16, 5 (2006): 519-536.

The purpose of this paper is that of linking security requirements for web services with security patterns, both at the architectural and the design level, obtaining in a systematic way a web services security software architecture that contains a set of security patterns, thus ensuring that the security requirements of the internet-based application that have been elicited are fulfilled. Additionally, the security patterns are linked with the most appropriate standards for their implementation.

Sabatier, D. & Lartigue, P. “The Use of the B Formal Method for the Design and Validation of the Transaction Mechanism for Smart Card Applications,” 348-368. FM ’99: World Congress on Formal Methods, Vol. I. Toulouse, France, Sept. 20-24, 1999. Berlin, Germany: Springer-Verlag, 1999. (Lecture Notes in Computer Science Vol. 1708.)

This document describes an industrial application of the B method in smart card applications. In smart card memory, data modification may be interrupted due to a card withdrawal or a power loss, the EEPROM memory may result in an unstable state and the values subsequently read, may be erroneous. The transaction mechanism provides a secure means for modifying data located in the EEPROM. As the security in smart card application is paramount, the use of the B formal method brings high confidence and provides mathematical proofs that the design of the transaction mechanism fulfils the security requirements.

Sawyer, P. & Kotonya, G. Ch. 2, “Software Requirements,” 9-34. Guide to the Software Engineering Body of Knowledge, Trial Version 1.00 (SWEBOK). Los Alamitos, CA: IEEE Computer Society, 2001. http://www.swebok.org/.

This document proposed a breakdown of the SWEBOK Software Requirements Knowledge area. The knowledge are is concerned with the acquisition, analysis, specification, validation and management of software requirements. It is widely acknowledged within the software industry that software projects are critically vulnerable when activities are performed poorly. This has led to the widespread use of the term ‘requirements engineering’ to denote the systematic handling of requirements. This is the term we use in the rest of this document. Software requirements are one of the products of the requirements engineering process.

Schneier, B. Secrets and Lies: Digital Security in a Networked World. New York, NY: John Wiley & Sons, 2000.

Who can you trust? Try Bruce Schneier, whose rare gift for common sense makes his book Secrets and Lies: Digital Security in a Networked World both enlightening and practical. He's worked in cryptography and electronic security for years, and has reached the depressing conclusion that even the loveliest code and toughest hardware still will yield to attackers who exploit human weaknesses in the users. The book is neatly divided into three parts, covering the turn-of-the-century landscape of systems and threats, the technologies used to protect and intercept data, and strategies for proper implementation of security systems. Moving away from blind faith in prevention, Schneier advocates swift detection and response to an attack, while maintaining firewalls and other gateways to keep out the amateurs.

Newcomers to the world of Schneier will be surprised at how funny he can be, especially given a subject commonly perceived as quiet and dull. Whether he's analyzing the security issues of the rebels and the Death Star in Star Wars or poking fun at the giant software and e-commerce companies that consistently sacrifice security for sexier features, he's one of the few tech writers who can provoke laughter consistently. While moderately pessimistic on the future of systems vulnerability, he goes on to relieve the reader's tension by comparing our electronic world to the equally insecure paper world we've endured for centuries—a little smart-card fraud doesn't seem so bad after all. Despite his unfortunate (but brief) shill for his consulting company in the book's afterword, you can trust Schneier to dish the dirt in Secrets and Lies. - Rob Lightner [This text refers to an out of print or unavailable edition of this title.]

Sindre, G. & Opdahl, A. "Eliciting Security Requirements by Misuse Cases," 120-130. Proceedings of TOOLS Pacific 2000. Sydney, Australia, Nov. 20-23, 2000. Los Alamitos, CA: IEEE Computer Society Press, 2000.

Use case diagrams (L. Jacobson et al., 1992) have proven quite helpful in requirements engineering, both for eliciting requirements and getting a better overview of requirements already stated. However, not all kinds of requirements are equally well supported by use case diagrams. They are good for functional requirements, but poorer at e.g., security requirements, which often concentrate on what should not happen in the system. With the advent of e- and m-commerce applications, security requirements are growing in importance, also for quite simple applications where a short lead time is important. Thus, it would be interesting to look into the possibility for applying use cases on this arena. The paper suggests how this can be done, extending the diagrams with misuse cases. This new construct makes it possible to represent actions that the system should prevent, together with those actions which it should support.

Sindre, G.; Opdahl, S.; & Brevik, G. "Generalization/Specialization as a Structuring Mechanism for Misuse Cases." SREIS 2002, Second Symposium on Requirements Engineering for Information Security. Raleigh, NC, Oct. 16, 2002. Lafayette, IN: CERIAS, Purdue University, 2002.

Use cases are becoming increasing common in the early phases of requirements engineering, but they offer limited support for expressing security requirements. However, misuse cases can specify behavior not wanted in the system. This paper presents and builds on previous work on misuse cases here focusing on misuse case generalization as a feature for structuring a larger number of misuse cases in a specification.

Software Engineering Institute. Requirements Management – Software Capability Maturity Model (SW-CMM) V2.0 Draft. http://www.sei.cmu.edu/cmm/draft-c/c21rqm.html (1997).

The purpose of requirements management is to establish and maintain a common agreement between the customer and the software project regarding the customer's requirements that will be addressed by the software project. Requirements management involves establishing and maintaining the baseline of allocated requirements for the software project, reviewing the software project's plans, activities, and work products to ensure that they are consistent with the allocated requirements.

Software Engineering Institute. International Workshop on Requirements for High Assurance Systems, Essen, Germany, September 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

High assurance systems (HASs) are computer systems where compelling evidence is required that the system delivers its services in a manner that satisfies certain critical properties. Among the critical properties are security properties (i.e., the system prevents unauthorized disclosure, modification and withholding of sensitive information), safety properties (the system prevents unintended events that could result in death, injury, illness, or property damage), survivability properties (the system continues to fulfill its mission in the presence of attacks, accidents, or failures), fault-tolerant properties (the system guarantees a certain quality of service despite faults, such as hardware, workload, or environmental anomalies), and real-time properties (the system delivers its outputs within specified time intervals).

Software Engineering Institute. International Workshop on Requirements for High Assurance Systems, Kyoto, Japan, September 6, 2004. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/community/rhas-workshop/.

A high assurance system is a software-intensive system that must dependably deliver its services by exhibiting sufficient safety, security, survivability, reliability, robustness, and performance. Safety critical systems are those high assurance systems that can cause accidents resulting in significant harm to people, property, or the environment. For such systems, safety risks must be reduced to an acceptably low level. Thus, developers of such systems must eliminate or minimize hazards to avoid accidents and minimize the negative consequences of any accidents that do occur.

The goal of the RHAS 04 Workshop was to bring together in a set of small focused working groups researchers and practitioners from the fields of safety engineering and requirements engineering to exchange ideas and their experiences concerning the engineering of safety-related requirements.

Much research and development remains to be done on this important problem, and together researchers and practitioners need to identify and explore important subproblems and propose, formulate, and evaluate promising solutions. This third workshop on Requirements for High Assurance Systems was one of many forums that will allow new ideas to be proposed and discussed.

A set of topics was identified for discussion at the workshop. The accepted papers are each associated with at least one of the topics. The topics include the following:

- What types of safety-related requirements exist, and how do their differences affect the way they are elicited, analyzed, and specified?
- What are useful processes and techniques for engineering safety-related requirements?
- What kinds of tools are needed to support the engineering of safety-related requirements?

Son, S. H. & Chaney, C. "Supporting the Requirements for Multi-Level Secure and Real-Time Databases in Distributed Environments," 73-91. IFIP '98. Vienna, Austria, Aug. 31-Sept. 4, 1998. Chapman & Hall, 1998.

Conflicts in database systems with both real-time and security requirements can sometimes be unresolvable. We attack this problem by allowing a database to have partial security in order to improve on real-time performance when necessary. By our definition, systems that are partially secure allow security violations between only certain levels. We present the ideas behind a specification language that allows database designers to specify important properties of their database at an appropriate level. In order to help the designers, we developed a tool that scans a database specification and finds all unresolvable conflicts. Once the conflicts are located, the tool takes the database designer through an interactive process to generate rules for the database to follow during execution when these conflicts arise. We briefly describe the BeeHive distributed database system, and discuss how our approach can fit into the BeeHive architecture.

Soo Hoo, Kevin; Sudbury, Andrew W.; & Jaquith, Andrew R. "Tangible ROI through Secure Software Engineering." Secure Business Quarterly 1, 2 (Fourth Quarter 2001). http://www.mudynamics.com/assets/files/Tangible%20ROI%20Secure%20SW%20Engineering.pdf.

Securely engineering software to proactively fix problems has a concrete value. Research proves that the return on investment for reviewing security in the design phase can be up to 21 percent.

Toval, A.; Nicolas, J.; Moros, B.; & Garcia, F. “Requirements Reuse for Improving Systems Security: A Practitioner’s Approach.” Requirements Engineering 6, 4 (January 2002): 205-219.

> We present a practical method to elicit and specify the system and software requirements, including a repository containing reusable requirements, a spiral process model, and a set of requirements documents templates. The method is focused on the security of information systems and, thus, the reusable requirements repository, contains all the requirements taken from MAGERIT, the Spanish public administration risk analysis and management method, which conforms to ISO 15408 common criteria framework. Any information system including these security requirements must therefore pass a risk analysis and management study performed with MAGERIT. The requirements specification templates are hierarchically structured and are based on IEEE standards. Finally, we show a case study in a system of our regional administration aimed at managing state subsidies.

Thayer, R. & Dorfman, M. Software Requirements Engineering, 2nd ed. Los Alamitos, CA: IEEE Computer Society Press, 1997 (ISBN 0-8186-7738-4).

> This new edition describes current best practices in requirements engineering with a focus primarily on software systems but also on systems that may contain other elements such as hardware or people. The text consists of original papers, written by experts in their fields, plus reprints of survey articles on many aspects of requirements engineering. The book begins with an introduction to current issues and the basic terminology of the soft ware requirements engineering process. In addition, the text covers the five basic phases of software requirements engineering: elicitation, analysis, specification, verification, and management.

Trammell, C. J. “Quantifying the Reliability of Software: Statistical Testing Based on a Usage Model,” 208-218. Proceedings of the Second IEEE International Symposium on Software Engineering Standards. Montreal, Quebec, Canada, August 21-25, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.

When a population is too large for study, as is the case for all possible uses of a software system, a statistically correct sample must be drawn as a basis for inferences about the population. In statistical testing of software based on a Markov chain usage model, the rich body of analytical results available for Markov chains provides numerous insights that can be used in test planning. Further, the connection between Markov chains and operations research techniques permits a Markov usage model to be expressed as a system of constraints, with mathematical programming used to generate the optimal model for a particular objective function. Since a software usage model is based on the specification, all analyses may be performed early in the development cycle and used as a quantitative basis for management decisions. These techniques have been reduced to engineering practice and used in large projects by IBM, Ericsson, all branches of the US military, and others. In this paper, statistical experiments, Markov models, and optimization techniques are shown to provide a sound theoretical and practical basis for quantifying the reliability of software.

Vetterling, M.; Wimmel, G.; & Wisspeintner, A. "Secure Systems Development Based on the Common Criteria: The PalME Project," 129-138. Proceedings of SIGSOFT 2002/FSE-10. Nov. 18-22, 2002, Charleston, SC. New York, NY: Association for Computing Machinery, 2002.

Security is a very important issue in information processing, especially in open network environments like the Internet. The Common Criteria (CC)is the standard requirements catalogue for the evaluation of security critical systems. Using the CC, a large number of security requirements on the system itself and on the system development can be defined. However, the CC does not give methodological support. In this paper, we show how integrate security aspects into the software engineering process. The activities and documents from the Common Criteria are tightly intertwined with the system development, which improves the quality of the developed system and reduces the additional cost and effort due to high security requirements. For modeling and verification of critical parts of the system, we use formal description techniques and model checking (supported by the graphical CASE tool AutoFocus, which increases both the understanding of the system specification and the system's reliability. We demonstrate our ideas by means of a case-study, the PalME project—an electronic purse application for Palm handhelds.

Viega, J. & McGraw, G. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001 (ISBN: 020172152X).

Cuts to the heart of computer security to help you get security right the first time. Your first step toward building more secure software. Provides expert perspectives and techniques to help you ensure the security of essential software.

From the Back Cover:

"This book is useful, practical, understandable, and comprehensive. The fact that you have this book in your hands is a step in the right direction. Read it, learn from it. And then put its lessons into practice."

- From the Foreword by Bruce Schneier, CTO, Counterpane, and author of Secrets and Lies
- "A must-read for anyone writing software for the Internet."
- Jeremy Epstein, Director, Product Security and Performance, webMethods
- "This book tackles complex application security problems like buffer overflows, race conditions, and applied cryptography in a manner that is straightforward and easy to understand. This is a must for any application developer or security professional."
- Paul Raines, Global Head of Information Risk Management, Barclays Capital

Most organizations have a firewall, antivirus software, and intrusion detection systems, all of which are intended to keep attackers out. So why is computer security a bigger problem today than ever before? The answer is simple--bad software lies at the heart of all computer security problems. Traditional solutions simply treat the symptoms, not the problem, and usually do so in a reactive way. This book teaches you how to take a proactive approach to computer security.

Building Secure Software cuts to the heart of computer security to help you get security right the first time. If you are serious about computer security, you need to read this book, which includes essential lessons for both security professionals who have come to realize that software is the problem, and software developers who intend to make their code behave. Written for anyone involved in software development and use--from managers to coders--this book is your first step toward building more secure software. Building Secure Software provides expert perspectives and techniques to help you ensure the security of essential software. If you consider threats and vulnerabilities early in the development cycle you can build security into your system. With this book you will learn how to determine an acceptable level of risk, develop security tests, and plug security holes before software is even shipped.

Inside you'll find the ten guiding principles for software security, as well as detailed coverage of:

- Software risk management for security
- Selecting technologies to make your code more secure
- Security implications of open source and proprietary software
- How to audit software
- The dreaded buffer overflow
- Access control and password authentication
- Random number generation
- Applying cryptography
- Trust management and input
- Client-side security
- Dealing with firewalls

Only by building secure software can you defend yourself against security breaches and gain the confidence that comes with knowing you won't have to play the "penetrate and patch" game anymore. Get it right the first time. Let these expert authors show you how to properly design your system; save time, money, and credibility; and preserve your customers' trust.

Whitten, A. & Tygar, J. “Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0.” Proceedings of the Usenix Security Symposium. Usenix Association, 1999.

User errors cause or contribute to most computer security failures, yet user interfaces for security still tend to be clumsy, confusing, or near-nonexistent. Is this simply due to a failure to apply standard user interface design techniques to security? We argue that, on the contrary, effective security requires a different usability standard, and that it will not be achieved through the user interface design techniques appropriate to other types of consumer software. To test this hypothesis, we performed a case study of a security program which does have a good user interface by general standards: PGP 5.0. Our case study used a cognitive walkthrough analysis together with a laboratory user test to evaluate whether PGP 5.0 can be successfully used by cryptography novices to achieve effective electronic mail security. The analysis found a number of user interface design flaws that may contribute to security failures, and the user test demonstrated that when our test participants were given 90 minutes in which to sign and encrypt a message using PGP 5.0, the majority of them were unable to do so successfully. We conclude that PGP 5.0 is not usable enough to provide effective security for most computer users, despite its attractive graphical user interface, supporting our hypothesis that user interface design for effective security remains an open problem. We close with a brief description of our continuing work on the development and application of user interface design principles and techniques for security.

Xie, N.; Mead, N. R.; Chen, P.; Dean, M.; Lopez, L.; Ojoko-Adams, D.; & Osman, H. SQUARE Project: Cost/Benefit Analysis Framework for Information Security Improvement Projects in Small Companies (CMU/SEI-2004-TN-045). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tn045.html.

Many companies rely on historical data to build predictability models for cost/benefit justification of future projects. Unfortunately, for small companies, which generally do not have a process for collecting security data, the costs and the benefits of information security improvement projects have been very difficult to estimate and justify. In addition, detailed attack data are simply not available to be used as references in cost estimations. Given these difficulties, many small companies choose to ignore entirely the security vulnerabilities in their systems, and many suffer the consequences of security breaches and significant financial loss. Small companies that do implement security improvement projects often have problems understanding the cost structures of their improvement initiatives and how to translate risk exposures into costs that can be passed on to their customers.

To deal with the aforementioned problems, this report describes a general framework for hierarchical cost/benefit analysis aimed at providing acceptable estimations for small companies in their information security improvement projects. The framework classifies misuse cases into categories of threats for which nationally surveyed risks and financial data are publicly available. For each category of threats, costs, benefits, baseline risks, and residual risks are estimated. The framework then generates all permutations of possible solutions and analyzes the most optimal approach to maximize the value of security improvement projects. The framework analyzes the problems from five dimensions: Total Implementation Costs, Total System Value, Net Project Value, Benefit/Cost Ratio, and Risk Exposures. The final proposed system will be derived from the comparisons of these dimensions, taking into consideration each company's specific situation.

This report is one of a series of reports resulting from research conducted by the System Quality Requirements Engineering (SQUARE) Team as part of an independent research and development project of the Software Engineering Institute.