Experiences in Eliciting Security Requirements

Dr. Nancy R. Mead
CERT, Software Engineering Institute

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.

A largely neglected area in requirements engineering is that of elicitation methods for security requirements. Many organizations, if they use an elicitation method at all for security requirements, use one that they have previously used for ordinary, functional (end-user) requirements. Alternatively, they may decide to use a brainstorming approach. Such methods are usually not oriented towards security requirements and do not result in a consistent and complete set of security requirements.

Carnegie Mellon University (CMU) graduate students, working with me, selected and applied several elicitation methods in a series of case studies [1]. In this article, I describe a trade-off analysis that we used to select a suitable requirements elicitation method and present results detailed from a case study of one method and a series of two other methods, used in a series of case studies. While results may vary from one organization to another, the discussion of our selection process and the example results should apply to all.

Elicitation Methods

The following is a sample of methods that could be considered for eliciting security requirements.

Misuse Cases

Misuse cases apply the concept of a negative scenario – that is, a situation that the system's owner does *not* want to occur – in a use-case context. For example, business leaders, military planners, and game players are familiar with analyzing their opponents' best moves as identifiable threats. By contrast, use cases generally describe behavior that the system or entity owner wants the system to show [2]. Use-case diagrams have proven quite helpful for the elicitation of requirements.

Soft Systems Methodology (SSM)

SSM deals with problem situations in which there is a high social, political, and human activity component [3]. The SSM can deal with *soft problems* that are difficult to define, rather than *hard problems* that are more technology oriented. Examples of *soft* problems are how to deal with homelessness, how to manage disaster planning, and how to improve Medicare. Eventually, technology-oriented problems may emerge from these *soft* problems, but much more analysis is needed to get to that point.

Quality Function Deployment (QFD)

QFD is an overall concept that provides a means of translating customer requirements into the appropriate technical requirements for each stage of product development and production [4]. The distinguishing attribute of QFD is the focus on customer needs throughout all product development activities. By using QFD, organizations can promote teamwork, prioritize action items, define clear objectives, and reduce development time.

Controlled Requirements Expression (CORE)

CORE [5, 6] is a requirements-analysis and specification method that clarifies the user's view of the services to be supplied by the proposed system. In CORE, the requirements specification is created by the user and the developer, not one or the other. The problem to be analyzed is defined and broken down into user and developer viewpoints. Information about the combined set of viewpoints is then analyzed. The last step of CORE deals with constraints analysis such as the limitations imposed by that system's operational environment in conjunction with some degree of performance and reliability investigation.

Issue-Based Information Systems (IBIS)

IBIS, developed by Horst Rittel, is based on the principle that the design process for complex problems, which Rittel terms *wicked* problems, is essentially an exchange among the stakeholders in which they bring their personal expertise and perspective to the resolution of design issues [7]. Any problem, concern, or question can be an issue and may require discussion and resolution in order for the design to proceed. The IBIS model centers on the discussion and resolution that is an integral part of the design process.

Joint Application Development (JAD)

JAD [8] is specifically designed for the development of large computer systems. The goal of JAD is to involve *all* stakeholders in the design phase of the product via highly structured and focused meetings. In the preliminary phases of JAD, the requirements-engineering team is tasked with fact finding and information gathering. Typically, the outputs of this phase – as applied to security requirements elicitation – are security goals and artifacts. The actual JAD session is then used to validate this information by establishing an agreed-upon set of security requirements for the product.

Feature-Oriented Domain Analysis (FODA)

FODA is a domain analysis and engineering method that focuses on developing reusable assets [9]. By examining related software systems and the underlying theory of the class of systems they represent, domain analysis can provide a generic description of the requirements of that class of systems in the form of a domain model, and a set of approaches for their implementation.

The FODA method was founded on two modeling concepts: abstraction and refinement. Abstraction is used to create domain models, as described above, from the specific applications in the domain. Specific applications in the domain are developed as refinements of the domain

models. The example domain used in the report [9] is that of window-management systems.

Critical Discourse Analysis (CDA)

CDA uses sociolinguistic methods to analyze verbal and written discourse [10]. In particular, CDA can be used to analyze requirements elicitation interviews and to understand the narratives and *stories* that emerge during requirements elicitation interviews.

Accelerated Requirements Method (ARM)

ARM is a facilitated requirements elicitation and description activity process [11]. Overall, there are three phases of the process: Preparation phase, Facilitated Session phase, and Deliverable Closure phase. The ARM process is similar to JAD, but it also has certain significant differences with respect to the baseline JAD method. In the ARM process, the facilitators are content-neutral, the group dynamic techniques used are different from those used in JAD, the brainstorming techniques used are different, and the requirements are recorded and organized using different conceptual models.

Evaluation Criteria

The following are example evaluation criteria (project participants need to have a common understanding of what they mean in order to use them in selecting an elicitation method):

- **Adaptability.** The method can be used to generate requirements in multiple environments. For example, the elicitation method works equally as well with a software package that is near completion as with a project in the planning stages.
- **Computer-aided software engineering (CASE) tool.** The method includes a CASE tool. (The Software Engineering Institute [SEI] defines a CASE tool as *a computer-based product aimed at supporting one or more software engineering activities within a software development process* [1].)
- **Stakeholder acceptance.** The stakeholders are likely to agree on the elicitation method in analyzing their requirements. For example, the method is not too invasive in a business environment and can be implemented in a reasonable amount of time.
- **Easy implementation.** The elicitation method is not overly complex and can be properly executed easily.
- **Graphical output.** The method produces readily understandable visual artifacts.
- **Quick implementation.** The requirements engineers and stakeholders can fully execute the elicitation method in a reasonable length of time.
- **Shallow learning curve.** The requirements engineers and stakeholders can fully comprehend the elicitation method within a reasonable length of time.
- **High maturity.** The elicitation method has experienced considerable exposure and analysis in its vetting by the requirements engineering community.
- **Scalability.** The method can be used to elicit the requirements of projects of different sizes, from enterprise-level systems to small-scale applications.

	Misuse Cases	SSM	QFD	CORE	IBIS	JAD	FODA	CDA	ARM
Adaptability	3	1	3	2	2	3	2	1	2
CASE Tool	1	2	1	1	3	2	1	1	1
Stakeholder Acceptance	2	2	2	2	3	2	1	3	3
Easy Implementation	2	2	1	2	3	2	1	1	2
Graphical Output	2	2	1	1	2	1	2	2	3
Quick Implementation	2	2	1	1	2	1	2	2	3
Shallow Learning Curve	3	1	2	1	3	2	1	1	1
High Maturity	2	3	3	3	2	3	2	2	1
Scalability	1	3	3	3	2	3	2	1	2
Total Score	**18**	**18**	**17**	**16**	**22**	**19**	**14**	**14**	**18**

3 = Very Good, 2 = Fair, 1 = Poor

Table 1: *Comparison of Elicitation Methods*

Ranking Against the Criteria

The elicitation methods can be ranked against the criteria using a tabular form. In Table 1, we have filled in the values that the student team decided on for the sample methods. Each method was rated according to the desired features, and the rankings were simply added to provide a summary result. A weighted average could also have been used if some features were considered to be more important than others. For example, availability of a CASE tool might be more important than graphical output. A typical weighting scheme could consider criteria to be *essential* with weight 3, *desirable* with weight 2, and *optional* with weight 1. This sort of evaluation is subjective, particularly since the students worked under time constraints and did not have prior experience with this, so results may vary. Each organization or project should develop its own comparison criteria and its own ratings.

In our case studies, we decided to use JAD, ARM, and IBIS on three different projects. These three methods were subjectively ranked to be the most suitable candidates for the case studies, given our constraints. The student team's allotted time was constrained, since this was a one-semester project. It was also the case, however, that the clients had limited time to devote to this exercise. Therefore, time constraints are mentioned several times.

It is important to note that although students did the elicitation, the projects studied were real industry and government projects, not software projects developed by students in an academic setting. It is also possible that a combination of methods may work best. This should be considered as part of the evaluation process.

Security Requirements Elicitation Results

In this section, we present brief results for IBIS and JAD and detailed results for ARM. Detailed results for all three methods can be found in the Requirements Engineering section of the BuildSecurityIn Web site [12] and in the case study report [1].

Brief Results for IBIS

The effectiveness of IBIS in eliciting security requirements depends on the quality of the interview questions. To the greatest extent possible, the scope of questions must cover the entire range of security requirements that could possibly involve the system. We found that the interviewer must be persistent in encouraging the stakeholders to explain their rationale when proposing a solution to an issue. By explaining why they have chosen such a position, the stakeholders can naturally discuss the pros and cons among themselves. In addition to proper question selection, we found that the success of IBIS is directly proportional to the variety and level of participation of stakeholders in the project. In fact, in our case study,

IBIS worked best when different stakeholders presented opposing viewpoints, which is common in large-scale projects. The Compendium software tool associated with IBIS was easy for the student team to use and effective in generating IBIS maps. To avoid displaying extremely large maps (which our stakeholders found difficult to read), we recommend exploiting the nested maps feature in Compendium. This feature enables the user to hide some of the lower-level details of the maps by nesting them inside other map elements, while maintaining the ability to drill down into the details if requested. In fact, a comment received from the stakeholders indicated that such a hierarchical map structure would have been more beneficial in handling some of the larger maps.

Brief Results for JAD

Due to time constraints, we did not define the work flow, data elements, screens, and reports of the project, so the JAD method turned out to be very similar to an unstructured interview process. Although unstructured interviews were used in an earlier case study, we did not attempt to do a direct comparison of the JAD results with those earlier case study results. In essence, the team just asked the stakeholders some questions about the project. Thus, the team did not use the full capability of the JAD method, which may have biased the results. The JAD session phase was designed for developing functional (end user) requirements; there was no specific way to discuss quality requirements such as security. Therefore, the team spent a lot of time researching other methods to assist in obtaining better security requirements during the JAD session. The team suggests that JAD be used with an additional method to deal with quality requirements.

Detailed Results for ARM

Results obtained using ARM on a government project are described below. This is not a military project, but the security concerns, such as access control, are similar to the typical security concerns of military projects. ARM is designed to elicit, categorize, and prioritize security requirements. As noted earlier, the ARM method includes three phases.

Preparation Phase

As the name implies, this phase is used to prepare for the Session phase. There are six steps in the Preparation phase:

1. Define goals, objectives, and project success criteria (PSC) of the project.
2. Define objectives and preliminary scope of the session.
3. Establish partitions and identify participants.
4. Determine environmental and logistical aspects.
5. Establish expectations for participants.
6. Communicate with participants.

One way to obtain this information is to use a feedback form composed of questions for the stakeholders. The list of questions can be found in the case study report [1]. The stakeholders should be given a few business days to complete and return the form. In the meantime, the requirements engineering team can prepare a memorandum containing goals, objectives, PSC, preliminary scope, partition definitions, participants, and logistic arrangements. Participants should read the memorandum before the Session phase to understand the content, expectation, and goals of the method. The overall goal of the memorandum is to increase the quality of the Session phase.

Results obtained using ARM on a government project are described here. This is not a military project, but the security concerns, such as access control, are similar to the typical security concerns of military projects.

Depending on the results of the stakeholders' feedback forms, another meeting with the stakeholders may be necessary before beginning the Session phase.

Session Phase

The Session phase is the heart of the ARM process. It includes six steps:

1. Executive sponsor commentary.
2. Scope closure.
3. Brainstorm, organize, and name (BON).
4. Details.
5. Prioritization.
6. Participant feedback.

Before the Session phase meeting, logistical arrangements should be made to ensure that the meeting goes smoothly. The detailed list of logistical items can be found in the case study report [1].

Executive Sponsor Commentary. This step allows executive sponsors to provide introductory remarks to the participants regarding the planned session. Depending on the project organization, this step may or may not be necessary.

Scope Closure. The purpose of this step is to define what is in or out of scope. When eliciting security requirements, participants might need to familiarize themselves with security issues ahead of time to make this determination.

BON. The BON step provides an efficient way to elicit the candidate requirements from participants. The requirements engineering team can start by asking the participants the *focus question*, which should be crafted to tie to the goals, objectives, and scope of the project together. For example: *An important security requirement of the beta application is…* Based on their professional experience and security knowledge, the participants are then asked to write down seven important security requirements within seven minutes.

Next, the participants are asked to write their top three or four security requirements on cards within three minutes. The requirements engineering team then collects the cards and displays the candidate security requirements. The candidate security requirements produced in this example are listed in the following 24 initial requirements produced in ARM:

1. The ability to securely transmit data to remote sources.
2. The preservation of data integrity.
3. The enforcement and usability of an access control system.
4. Manageable security (and not hinder business where possible).
5. A strong, reliable authentication process.
6. Private information (from the outside world).
7. Consistent application program interfaces (APIs).
8. Data integrity.
9. Authentication and access control.
10. Strong authentication.
11. Risk reduction or elimination of risk of inappropriate behavior.
12. Granular access to data for users (operators) and customers.
13. Accountability (who did what, when, how...).
14. Integrity (assurance in data protection and validity).
15. Indelibility (deletions and retractions are noted/logged).
16. Integrity.
17. Access control.

18. Confidentiality (encryption, etc.).
19. Partitioned data store (public read only and private read/write).
20. Selectively secure communication with outside entities.
21. Segmented disclosure representation and support.
22. Role-based restricted views/edit/ action access (e.g., summary report information, public for particular people).
23. 24/7 availability via remote authenticated access and secure.
24. Key action audit (e.g., attribution of who/from where the publish button was pressed, what changes were made).

In the Organize step, all the participants review the candidate security requirements generated during the brainstorming session to see whether any duplicate or inadequate security requirements were included. Then the participants discuss what they think are important requirements. This step provides an opportunity for the participants to share their security concerns about the project. After a period of discussion and debate, they delete candidate security requirements that are viewed as redundant or inappropriate.

The participants removed requirements 1, 2, 5, 9, 14, 16, and 17. The remaining requirements after initial eliminations are:

3. The enforcement and usability of an access control system.
4. Manageable security (and not hinder business where possible).
6. Private information (from the outside world).
7. Consistent APIs.
8. Data integrity.
10. Strong authentication.
11. Risk reduction or elimination of risk of inappropriate behavior.
12. Granular access to data for users (operators) and customers.
13. Accountability (who did what, when, how...).
15. Indelibility (deletions and retractions are noted/logged).
18. Confidentiality (encryption, etc.).
19. Partitioned data store (public read only and private read/write).
20. Selectively secure communication with outside entities.
21. Segmented disclosure representation and support.
22. Role-based restricted views/edit/ action access (e.g., summary report information, public for particular people).
23. 24/7 availability via remote authenticated access and secure.
24. Key action audit (e.g., attribution of who/from where the publish button was pressed, what changes were made).

In the Name step, the participants are instructed to group the selected security requirements and create names for each group. In this example, security requirements, groups, and names were generated together and descend in order of importance from A-F (as shown in Figure 1, page 18). The security requirements are categorized into six groups, each containing between one and four security requirements. This step can result in addition or deletion of requirements. The following are the grouped requirements contained in each:

- **Group A: Confidentiality:** Information must be kept private from the outside world; communication with outside entities must be selectively secured.
- **Group B: Access Control:** Role-based restricted views/edit/action access (e.g., summary report information, public for particular people); enforcement and usability of an access control system; granular access to data for users (operators) and customers; segmented disclosure support and representation.
- **Group C: Data Integrity:** Partitioned data store (public read only and private read/write); indelibility.
- **Group D: Manageability:** Accountability; key action audit (e.g., attribution of who/from where the publish button was pressed and what changes were made); auditing capabilities.
- **Group E: Usability:** Security must be manageable and not hinder business (where possible); must be available 24/7 via remote authenticated access; must have consistent APIs; must reduce or eliminate risks of inadvertent behavior.
- **Group F: Authentication:** Strong authentication.

Details: Benefits, Proof, Assumptions, Issues, and Action Items. In Step 4, the participants are asked to evaluate each requirement using the following 10 questions:

1. Is the candidate requirement a fragment or duplicate of anything that has already been discussed?
2. According to the contributor and the group, is the candidate requirement fragment in scope?
3. Would you like to change the title?
4. If you had this capability, how would it help the business?
5. What will you consider acceptable evidence that the envisioned capability

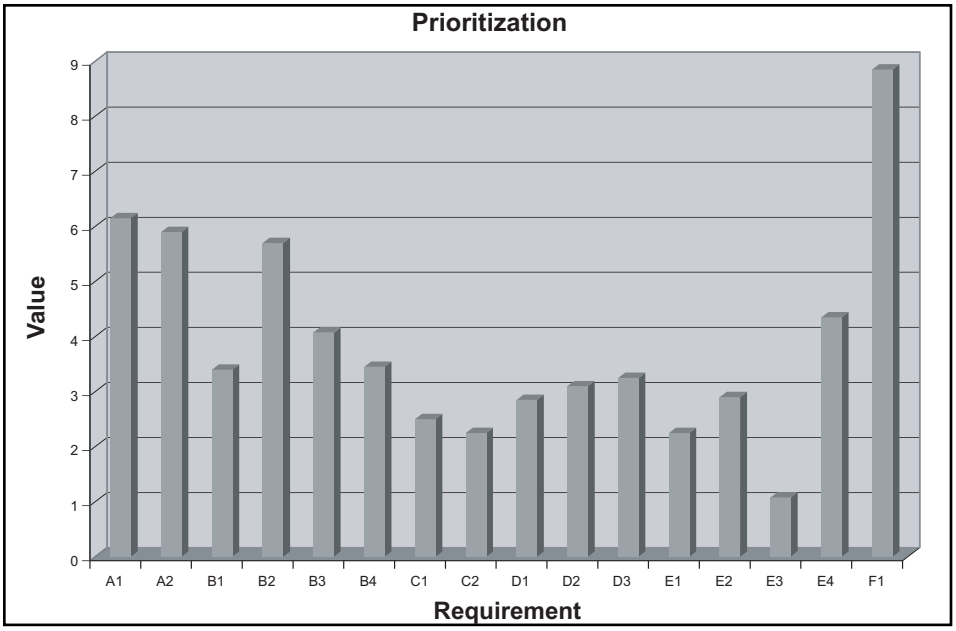

Figure 1: *Ranked Score of the Requirements*

has been successfully delivered to the business?
6. Are there any special constraints on the requirement?
7. Are there any assumptions made regarding the requirement?
8. What are the remaining issues and actions items for the requirement?
9. Are there any related notes or comments?
10. Is there anything that needs to be clarified by the supplier of the requirement?

In this case, the security requirements were reviewed collectively, not individually.

Prioritization. In the BON step of ARM, the participants generate the candidate security requirements of their project, then modify and refine the projected security requirements in the Details step to ensure that the requirements are unambiguous, clear, and concise.

The Prioritization phase of the ARM method begins with the requirements engineering team providing instructions to guide participants to label each requirement as either *A*, *B*, or *C*, where *A* stands for most important, *B* stands for very important, and *C* stands for important. The rankings are to be assigned equally across the security requirements.

After the session concludes, the scores are calculated. First, the requirements engineering team substitutes the rankings *A*, *B*, and *C* with numeric values 9, 3, and 1, respectively. Then, the team calculates the average score of each requirement. The results are shown in Figure 1.

These are the final requirements in priority order:

1. The system shall utilize cryptographically strong authentication.
2. The information in the system must be kept private from unauthorized users.
3. The system shall implement selectively secure communication with outside entities.
4. The system shall utilize and enforce an access control system.
5. The system will attempt to reduce or eliminate risks of inadvertent behavior.
6. The system shall provide granular access to data for users (operators) and customers.
7. The system shall provide role-based, restricted view, edit, and action access (e.g., summary report information, public for particular people).
   (tied with)
   The system shall represent and support segmented disclosure.
8. The system shall implement auditing capabilities.
9. The system shall provide accountability of users' actions.
   (tied with)
   The system will be available 24/7 via remote authenticated access.
10. The system shall maintain a partitioned data store that is public read only and private read/write.
    (tied with)
    The system shall implement a key action audit (e.g., attribution of who pressed the publish button and from where, and what changes were made).
11. The system shall implement indelibility.
    (tied with)
    Where possible, the system's security features must be manageable and not hinder business.
12. The system shall expose consistent APIs to developers.

Based on the result of the prioritization, the participants can then develop a plan to effectively implement their security requirements.

Stakeholders' Feedback. In the final portion of the Session phase, the team requested that the participants fill out a feedback form that was used to collect information to improve the method. Example questions are similar to the following:

- What did you like or not like about the Session phase?
- What did you think was the most important part of the Session phase?
- What would you change about the Session phase?

Deliverable Closure

In this study, the set of stakeholders was relatively small and Deliverable closure took place informally at the Session phase.

Results Summary for ARM

Overall, ARM is an effective and rapid method of collecting requirements. By simply choosing the correct focus question, the process is easily adapted to elicit security requirements. Due to the large number of questions that must be asked for each requirement, we recommend enforcement of strict time management and proactive guidance of the discussions among the stakeholders.

Depending on the security expertise of the participants, the requirements engineering team may need to review some security concepts with the participants before the session begins.

ARM was developed for use in a commercial environment, and thus, may focus excessively on features.

Results Summary for All Elicitation Methods

ARM seemed better suited to elicitation of security requirements than either IBIS or JAD. JAD seemed more suited to end-user functional requirements and provided no specific way to discuss quality requirements such as security. We found that IBIS was effective for documenting complex decision-making discussions but did not provide a structured way of generating security requirements.

We later experimented with other prioritization methods [1], notably Analytic Hierarchy Process (AHP), which seemed to provide more systematic prioritization than ARM.

Future Plans

These case studies are part of the securi-

ty quality requirements engineering (SQUARE) project [13]. Current plans call for a comparison of SQUARE with other security requirements engineering methods, experimental combination of elicitation and prioritization methods (for example, combining ARM for elicitation with AHP for prioritization), development of supporting tools, and development of tutorial materials.◆

Acknowledgement

This work was part of a more extensive case study project by CMU graduate students Lydia Chung, Frank Hung, Eric Hough, and Don Ojoko-Adams [1]. This work is supported by the Army Research Office through grant number DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to CMU's CyLab.

References

1. Chung, L., et al. Security Quality Requirements Engineering (SQUARE): Case Study Phase III. Pittsburgh, PA: SEI, CMU, 2006 <www.sei.cmu.edu/publications/documents/06.reports/06sr003.html>.
2. Sindre, G., and A.L. Opdahl. "Eliciting Security Requirements by Misuse Cases." 37th International Conference on Technology of Object-Oriented Languages (Tools 37-Pacific 2000). Sydney, Australia, Nov. 20-23, 2000. Los Alamitos, CA: Institute of Electrical and Electronics Engineers Computer Society, 2000.
3. Checkland, P. Soft Systems Methodology in Action. Ontario, Canada: John Wiley & Sons, 1990.
4. QFD Institute. "Frequently Asked Questions About QFD." 2005 <www.qfdi.org/what_is_qfd/faqs_about_qfd.htm>.
5. Christel, M., and K. Kang. Issues in Requirements Elicitation. Pittsburgh, PA: SEI, CMU, 1992 <www.sei.cmu.edu/publications/documents/92.reports/92.tr.012.html>.
6. Systems Designers Scientific. CORE – The Method: User Manual. London, England: SD-Scicon, 1986.
7. Kunz, W., and H. Rittel. Issues as Elements of Information Systems. Berkeley: Institute of Urban and Regional Development, University of California, 1970 <www.iurd.ced.berkeley.edu/pub/WP-131.pdf>.
8. Wood, J., and D. Silver. Joint Application Development. 2nd ed. New York: John Wiley & Sons, 1995.
9. Kang, K.C., et al. Feature-Oriented Domain Analysis Feasibility Study. Pittsburgh, PA: SEI, CMU, 1990 <www.sei.cmu.edu/publications/documents/90.reports/90.tr.021.html>.
10. Schiffrin, D. Approaches to Discourse. Oxford, England: Blackwell Publishers Ltd, 1994.
11. Hubbard, R., Nancy R. Mead, and C. Schroeder. "An Assessment of the Relative Efficiency of a Facilitator-Driven Requirements Collection Process With Respect to the Conventional Interview Method." Proc. of the International Conference on Requirements Engineering. Los Alamitos, CA: IEEE Computer Society Press, 2000.
12. United States. Dept. of Homeland Security. BuildSecurityIn Portal. National Cyber Security Division <https://buildsecurityin.us-cert.gov/>.
13. Mead, N.R., et al. Security Quality Requirements Engineering (SQUARE) Methodology. Pittsburgh, PA: SEI, CMU, 2005 <www.sei.cmu.edu/publications/documents/05.reports/05tr009.html>.

About the Author

Nancy R. Mead, Ph.D., is a senior member of the technical staff in the Networked Systems Survivability Program at the SEI and is also a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at CMU. Her research interests are in the areas of information security, software requirements engineering, and software architectures. Mead has more than 100 publications and invited presentations. She has a doctorate in mathematics from the Polytechnic Institute of New York and bachelor and master degrees in mathematics from New York University.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213
E-mail: nrm@sei.cmu.edu