Ensuring Cost Efficient and Secure Software through Student Case Studies in Risk and Requirements Prioritization

Nancy R. Mead
Software Engineering Institute
nrm@sei.cmu.edu

Dan Shoemaker
University of Detroit Mercy
dshoemaker1@twmi.rr.com

Jeffrey Ingalsbe
Ford Motor Company
jingalsb@ford.com

Abstract

This paper presents a discussion of educational case studies used in security requirements assessment and requirements prioritization. Related to this, it introduces risk understanding as an added dimension to the requirements prioritization process. It should be self-evident that the final product should incorporate the requirements with the greatest value. Nevertheless, in a time when security is a preeminent concern it should also be clear that risk elements should also be considered. As such, activities to reconcile risk with value are always essential. However, since risk and value considerations are different, and sometimes opposed to each other, this paper presents a new process that will help decision makers reconcile these two factors within a single approach. This new process may also be incorporated into security requirements education and prioritization.

1. Identifying the right requirements

It is well documented that requirements engineering saves money. That is because, according to a number of studies [3, 5, 12, 17, 18, 26] to name a few) requirements concerns are the primary reason for projects to be significantly over budget, past schedule, or cancelled. Other studies have shown that reworking requirements defects on most software development projects costs 40 to 50 percent of total project effort, and the percentage of defects originating during requirements engineering is estimated at more than 50 percent [18, 20].

So it is important to build a sound requirements set. However, besides the classic cost and feasibility issues, it is also critical to understand the potential risks associated with implementing each requirement. That is because of the absolute need to ensure that the software that underpins our infrastructure is secure. Yet up to this point, it is estimated that the exploitation of unsecured software costs the U.S. economy an average of $60 billion dollars per year [24]. More importantly perhaps, the exploitation of a flaw in the software that underlies basic services like power and communication could cause a significant disaster [7].

Notwithstanding the national security concerns, there are also sound business reasons for developing secure software. An earlier study found that the return on investment when security analysis and secure engineering practices are introduced early in the development cycle ranges from 12 to 21 percent, with the highest rate of return occurring when the analysis is performed during application design [32]. Thus, the costs of poor security requirements argue that even a small improvement in this area would provide a high value.

2. The special case of security requirements

The concern comes from the fact that security requirements are typically developed independent of functional requirements in the requirements engineering process. Much of requirements engineering research and practice has addressed the capabilities that the system will provide. So, from the user's perspective, a lot of attention is paid to the functionality of the system. But little attention is given to what the system should *not* do.

As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are either not implemented, or are strapped on in a piecemeal and disaggregate fashion at the end of the process. Worse, requirements that might add functional value but which represent added risk are never considered from that perspective. For instance, in the requirements prioritization process, ease of use is typically assigned a much higher priority, or is often traded off

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

against, rigorous access control. This means that any adversary can more easily attack the target system.

Therefore, in addition to employing applicable software engineering methods, the organization also has to understand how risk considerations impact its development processes. This is essential because if risk considerations are not a formal part of the requirements elicitation and prioritization process, then security tends to be neglected during the build. That reason alone should justify the need to embed security considerations into the initial requirements elicitation and prioritization process.

3. Understanding requirements by understanding threats

A proper understanding of the threat picture allows the development team to do a better job of prioritizing requirements. Logically, the place to start when attempting to assess the security status of a given requirement is to understand all threats associated with it. That is because, without an adequate practical understanding of every conceivable way that a given requirement might be compromised, it is difficult to make a decision about its priority in the eventual product.

Threat understanding, is the role of threat modeling. Most threat-modeling methodologies share common themes. They emphasize systematic, repeatable processes with work products that enhance communication and highlight risk in a clear, easy to understand way for all stakeholders. They differ on several planes (scope, perspective, formality, and automation). The most popular threat modeling technique comes from Microsoft. It focuses on data flow and trust boundaries [33]. The Trike methodology [29] emphasizes an application's perspective while encouraging automation. Myagmar [22] focuses on an attacker's perspective but heavily emphasizes the role threat modeling plays in developing security requirements.

Threat models identify threats, but risk analysis is needed in order to understand the practical risk that each threat represents. At a minimum, risk analysis involves assessing the likelihood and impacts of each identified threat. That is because, without an adequate understanding of these two factors, it is hard to make a realistic decision about the consequences of any identified threat. An adequate risk analysis will ensure that all consequences of threat are factored into the requirements prioritization process.

4. Assessing security risks through educational case studies

There are many techniques for assessing software risk [45]; however, not all of them are useful for assessing security risks. In previous security requirements engineering student team case studies [42, 43] a number of risk assessment techniques were identified as potentially useful for identifying security risks after completing a literature review. This review by student teams examined the usefulness and applicability of eight risk assessment techniques:

1. General Accounting Office Model [38]
2. National Institute of Standards and Technology (NIST) Model [39]
3. NSA's INFOSEC Assessment Methodology [41]
4. Shawn Butler's Security Attribute Evaluation Method [34]
5. Carnegie Mellon's Vendor Risk Assessment and Threat Evaluation [37]
6. Yacov Haimes's Risk Filtering, Ranking, and Management Model [36]
7. Carnegie Mellon's Survivable Systems Analysis Method [40]
8. Martin Feather's Defect Detection and Prevention Model [35]

Each method was ranked in four categories:

1. Suitability for small companies
2. Feasibility of completion in the time allotted
3. Lack of dependence on historical threat data
4. Suitability in addressing requirements

The results of the ranking are shown in Table 1.

After averaging scores from the four categories, NIST's and Haimes's models were selected as useful techniques for security risk assessment. Brainstorming, attack tree, and misuse case documentation were used to identify potential threat scenarios. The two independent risk assessment analyses produced a useful risk profile for the educational case study using quantitative methods.

In this particular case study, the student team also identified a set of essential services and assets. This activity is not part of the risk techniques used, but nevertheless can be a beneficial exercise if enough architectural information already exists to support it. All findings from the risk assessment, along with the findings from the essential services and asset identification process, were used to help determine the priority level associated with each of the requirements.

5. Aligning and prioritizing value and risk: a prototype approach

AHP is a popular method for supporting decision making where multiple objectives are present [28, 15, 16]. This method uses a "pair-wise" comparison matrix to calculate the relative value and costs of security requirements. By using AHP, the requirements engineer can also confirm the consistency of the result. There are five steps in the AHP method:

1. Review candidate requirements for completeness.
2. Apply a pair-wise comparison method to assess the relative value of requirements.
3. Estimate the relative cost of implementing each candidate requirement.
4. Calculate each requirement's relative value and implementation cost, and plot each on a cost-value diagram.
5. Use the cost-value diagram as a map for analyzing the candidate requirement.

In prior student team case studies, stakeholders implemented the pair-wise comparison method of AHP [44]. The stakeholders were provided brief instructions for using the AHP methodology. Because there were 9 security requirements, the method should produce a matrix with 81 (9 x 9) cells. However, the participants needed to fill the upper half of the matrix only, and each requirement had a value of "1" when compared to itself. Consequently, each participant had to respond to 36 cells. (In general, a matrix with n requirements must have responses in n * (n-1) cells. This rule reduced the participants' workload in this step.) Involvement of a number of stakeholders in this process leads to a structured discussion of the results and helps to build consensus on the validity of the cost/value assessments. The requirements engineer highlighted the cells that required feedback from the stakeholders. A sample of the feedback is shown in Table 2.

Table 1. Ranking of assessment techniques

		Suitable for Small Companies	Feasible to Complete within Time Frame	Does Not Require Additional Data Collection	Suitable for Requirements	Average Score
Methodologies	GAO	2	4	2	2	2.50
	NIST	2	2	1	1	1.50
	NSA/IAM	3	3	2	2	2.50
	SAEM	4	4	4	4	4.00
	V-Rate	3	4	4	4	3.75
	Haimes	2	2	2	2	2.00
	SSA	2	2	2	4	2.50
	DDP/Feather	3	4	2	4	3.25

Table 2. Prioritization Feedback of Acme

	A	B	C	D	E	F	G	H	I	J
1		**SR-1**	**SR-2**	**SR-3**	**SR-4**	**SR-5**	**SR-6**	**SR-7**	**SR-8**	**SR-9**
2	**SR-1**	1	8	1/5	3	1	2	2	3	1
3	**SR-2**	1/8	1	1/5	1/7	1/7	1/7	1/7	1/9	1/9
4	**SR-3**	5	5	1	1	2	1	3	1	1
5	**SR-4**	1/3	7	1	1	1/2	1/2	3	1/2	1
6	**SR-5**	1	7	1/2	2	1	3	3	1	1/3

7	**SR-6**	1/2	7	1	2	1/3	1	1/3	1	1
8	**SR-7**	1/2	7	1/3	1/3	1/3	3	1	3	2
9	**SR-8**	1/3	9	1	2	1	1	1/3	1	1/6
10	**SR-9**	1	9	1	1	3	1	1/2	6	1

AHP uses a pair-wise comparison matrix to determine the relative value and cost between security requirements. An arbitrary entry in row i and column j of the matrix, labeled a_{ij}, indicates how much higher (or lower) the value/cost for requirement i is than that for requirement j. The value/cost is measured on an integer scale from 1 to 9.

To illustrate this, consider the nine security requirements in our case study (the requirements' final value is shown in Figure 1, and the requirements' final cost is shown in Figure 2). The value of each requirement is relative. That is, if the value of a requirement is 20%, this requirement is twice as important as the 10% value of another requirement. The sum of the scores of the requirements should always be 100%.

According to Figure 1, the three most valuable requirements are SR-3, SR-5, and SR-6. Together, they constitute 47% of the requirements' total value. The three least valuable requirements are SR-1, SR-4, and SR-8, which constitute 23% of the requirements' total value. Figure 2 shows that requirements SR-4, SR-7, and SR-9 are the three most expensive. Together, they constitute 72% of the requirements' total cost. The three least expensive requirements are SR-1, SR-2, and SR-3 that constitute 7% of the requirements' total cost.

The next step calculates the cost-value ratios for each requirement. The aim is to pinpoint the requirements that are most valuable and least expensive to implement based on:

- **high** value-to-cost ratio of requirement (larger than 2.0)
- **medium** value-to-cost ratio of requirement (between 2.0 and 0.5)
- **low** value-to-cost ratio of requirement (less than 0.5)

Using this, SR-1, SR-2, SR-3, SR-5, and SR-6 are identified as high priority and SR-4 and SR-7 are identified as low priority.

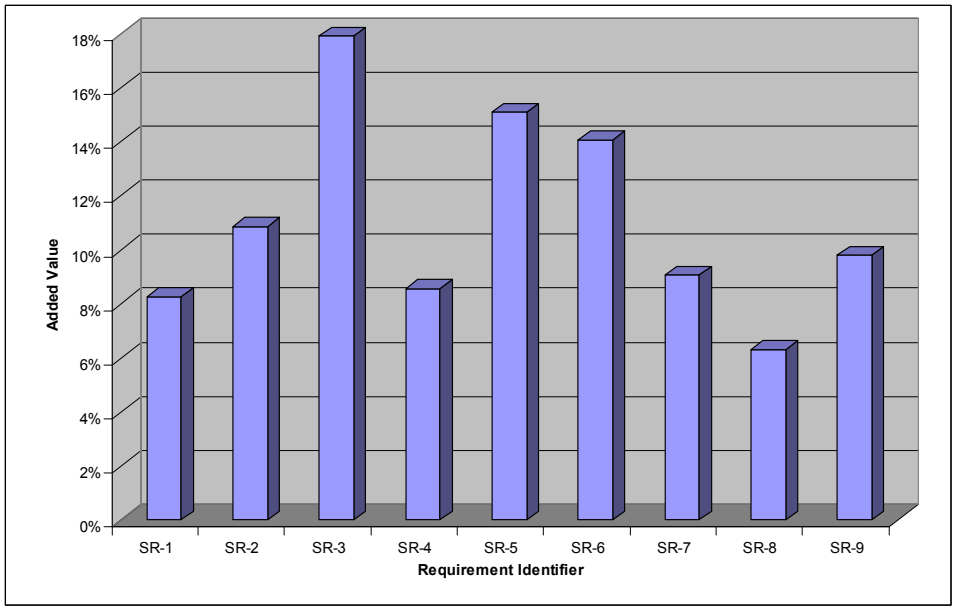

Figure 1. Value distribution of requirements

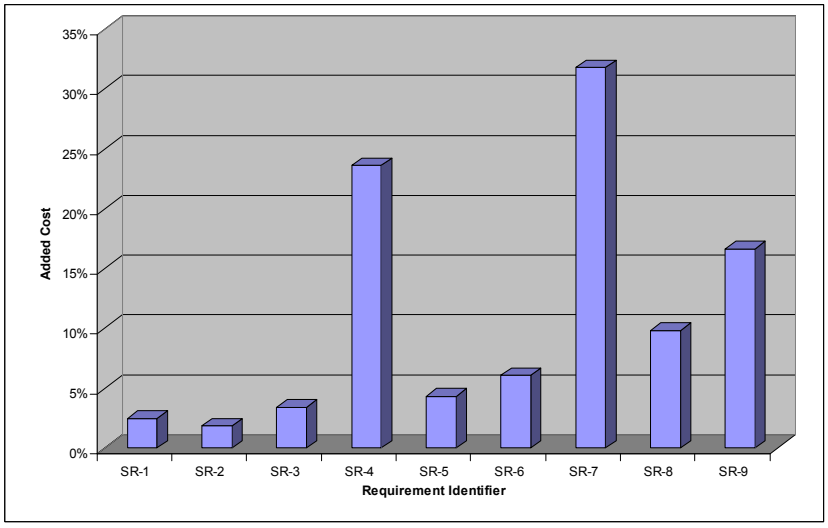

Figure 2. Cost distribution of requirements

6. Adding threat assessment to the prioritization process

AHP provides a good quantitative basis for making a decision about the relative priority of a given set of requirements. However, it does not factor in the additional dimension of risk. In order to do that all risks associated with each requirement have to be identified and assessed for likelihood and impact. As noted before, there are many ways of modeling risk. These days, threat modeling is also a popular alternative, which we illustrate in conjunction with AHP prioritization, by subjecting our sample set to a threat modeling exercise. A sample risk assessment form along these lines can be found at http://www.securedigitalsolutions.com/articles/Generic%20Risk%20Assessment%20Form.pdf .

Each threat associated with each requirement is identified through a threat model. That might produce such hazards as "subject to cross site scripting attacks", or "inadequate access controls". Each of these threats is then ranked on two factors, likelihood and impact. The ranking is expressed on a seven level Likert scale ranging from highest-to-lowest. The result would produce two outcomes likelihood (1-7) and impact (1-7). Since these factors have equal weight, in order to normalize them these two scores could then be multiplied together in order to obtain a single risk factor score ((L) likelihood * (I) impact = (R) risk) for each threat. Then all of the individual threat scores are multiplied to obtain a single threat index. The aim is to rank the requirement set by the relative level of threat associated with each requirement. In our example that might produce the ranking shown in Figure 4.

These are then ranked as very high risk (>200), high risk (>100), moderate risk (>50) low risk (>25) and no risk (<25). In this case, SR-7 is a very high-risk requirement. SR-6 and SR-4 are high-risk requirements. SR-1, SR-2, and SR-3 are moderate risk requirements and SR-5, SR-8 and SR-9 represent negligible risks

Then referring to the results of the AHP ranking it is possible to think about these requirements in a different way. SR-6, which was identified as a high priority, is also a high risk. Whereas, SR-1, SR-2 and SR-3, which are high priority items, are shown to only represent a moderate risk. Better still SR-5, which is a high priority requirement, is a low risk. And to make a further case against SR-4 and SR-7, which were identified as low priority requirements in the AHP analysis, these two requirements also represent the two greatest risks in the requirements set.

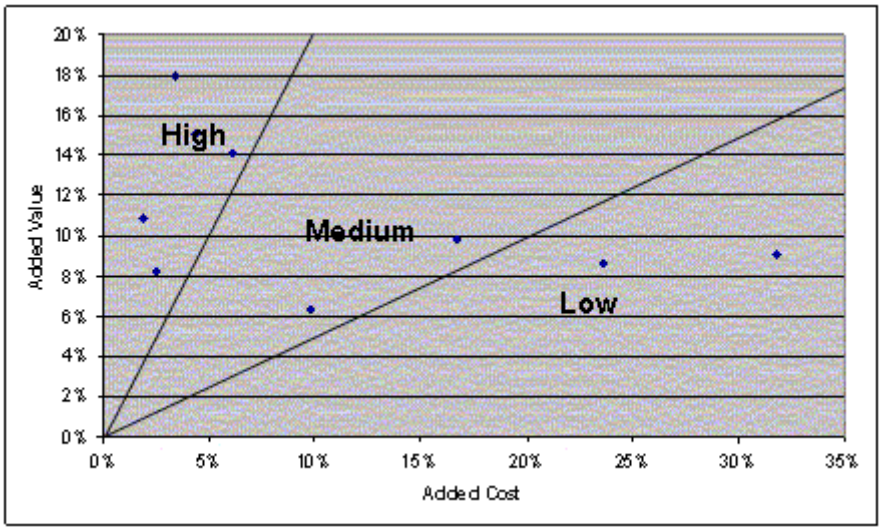

Figure 3. Cost-value diagram of requirements

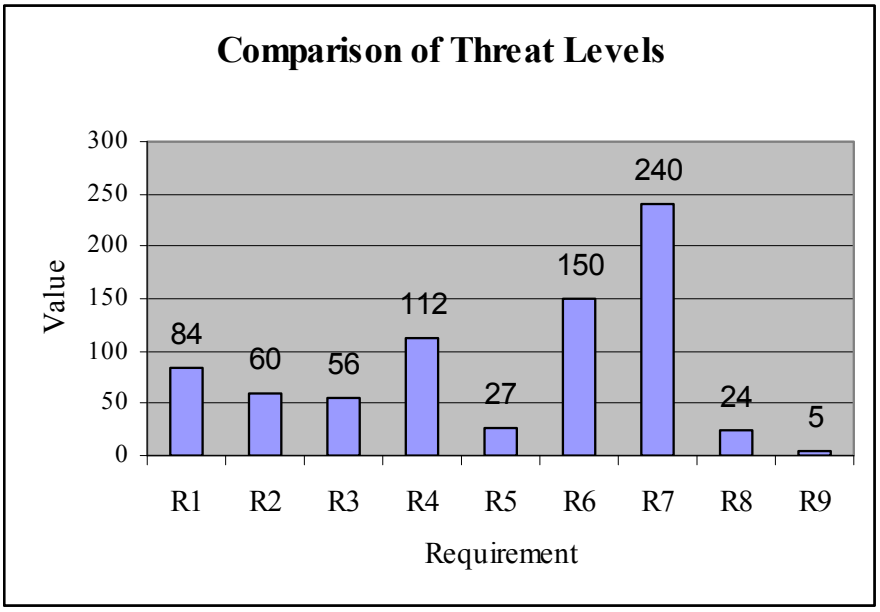

Figure 4. Comparative threat levels per requirement

That outcome changes our picture somewhat, in the sense that decision makers might want to revisit, or perhaps re-prioritize based on potential risk. This is particularly true in the case of SR-6. It might also suggest that the status of SR-1, which is approaching high-risk status, be revisited. At the same time, it is easy to justify the priority of SR-2, SR-3 and SR-5 based on their relative threat index. Finally, it is also easy to consider dropping SR-4 and SR-7 out of the set if resource constraints arise.

This modified approach has not yet been applied in our student case studies. One of the challenges we face is that students have an expectation for crisp results. They are not used to the idea that when they are doing research case studies, the results may be unpredictable. In our view, this makes these kinds of case studies more interesting and more challenging for the students, as well as helping our research.

7. Conclusions

The aim of this article was to introduce the potential additional value of threat modeling/ risk understanding to the requirements prioritization process. The information that these additional analysis processes provide permits decision makers

to undertake the construction process in such a way that the most critical functional and security requirements are always included in the final build. Exposing students to these processes can help them understand the sophisticated decisions that must be made in real development settings.

Moreover, that information can then be used by decision makers to make intelligent decisions about the amount of investment that they wish to make in their software. That will allow them to maximize their investment and thereby ensure both their profitability the as well as the overall safety and security of their organization. That will allow every member of the organization to be confident that their software assets provide the greatest value for the investment as well as the greatest security possible.

From the educational perspective, if we introduce students to sophisticated decision making processes in the real world, they will be better equipped to make such decisions when the time comes.

8. References

[1] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software*, vol. 20, no. 1, pp. 58-66, 2003.

[2] T. Benzel, "Integrating security requirements and software development standards," in *Proc. 12th National Computer Security Conf*. Fort Meade, MD: National Computer Security Center, 1989, pp. 435-458.

[3] B. Boehm and V. Basili, "Software defect reduction – Top 10 list," *IEEE Computer*, vol. 34, no. 1, pp. 135-137, Jan. 2001.

[4] J. Bailey, A. Drommi, J. Ingalsbe, N. Mead, and D. Shoemaker, "Models for Assessing the Cost and Value of Software Assurance," *Build Security In*, https://buildsecurityin.us-cert.gov/daisy/bsi/articles /knowledge/business/684.html

[5] J.J. Carr, "Requirements engineering and management: The key to designing quality complex systems," *The TQM Magazine*, vol. 12, no. 6, pp. 400-407, 2000.

[6] R.A. Clark, *Breakpoint*. New York: G.P Putnam and Sons, 2007.

[7] R.A. Clark and H.A. Schmidt, "A national strategy to secure cyberspace," The President's Critical Infrastructure Protection Board, Washington, DC, 2002.

[8] E. Colbert and D. Yu, "Costing Secure Systems Workshop Report," 21st Annual Forum on CoCoMo and Software Cost Modeling, University of Southern California, Center for Software Engineering, 2006.

[9] R. de Landtsheer and A. van Lamsweerde, "Reasoning about confidentiality at requirements engineering time." in *Proc. 10th European Software Engineering Conf*. held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, NY: ACM, 2005, pp. 41-49.

[10] K. Garg and V. Varma, "Security: Bridging the academia-industry gap using a case study," in *XIII Asia Pacific Software Engineering Conf. Proc*. New York, NY: IEEE Computer Society Press, 2006, pp. 485-492.

[11] P. Giorgini, H. Mouratidis, and N. Zannone, "Modeling Security and Trust with Secure Tropos," *Integrating Security and Software Engineering: Advances and Future Visions*. Hershey, PA: IGI Global, 2007, pp. 160-189.

[12] H. Hecht, and M. Hecht, "How reliable are requirements for reliable software?" *Software Tech News*, vol. 3, no. 4, 2000. Retrieved May 31, 2007 from http://www.softwaretechnews.com.

[13] S. Honiden, Y. Tahara, N. Yoshioka, K. Taguchi, and H. Washizaki, "Top SE: Educating superarchitects who can apply software engineering tools to practical development in Japan," in *Proc. 29th Int. Conf. on Software Engineering (ICSE'07)*. New York, NY: IEEE Computer Society, 2007, pp. 708-718.

[14] M. Howard and D. LeBlanc, *Writing Secure Code* (with CD-ROM). Microsoft Press, 2001.

[15] J. Karlsson, "Software Requirements Prioritizing," in *Proc. Second Int. Conf. on Requirements Engineering (ICRE'96)*. Colorado Springs, CO, April 15-18, 1996. Los Alamitos, CA: IEEE Computer Society, 1996, pp. 110–116.

[16] J. Karlsson, and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software* vol. 14, no. 5, Sept./Oct. 1997, pp. 67-74.

[17] S. Lauesen and O. Vinter, "Preventing requirement defects: An experiment in process improvement," Requirements Engineering, Volume 6, Number 1, February, 2001, pp. 37-50.

[18] T. McGibbon, "A business case for software process improvement revised," DoD Data Analysis Center for Software (DACS), Washington, DC, 1999.

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

[20] N.R. Mead and T. R. Stehney, II, "Security quality requirements engineering (SQUARE) methodology," paper presented at the meeting of the Software Engineering for Secure Systems (SESS05), ICSE 2005 Int. Workshop on

Requirements for High Assurance Systems, St. Louis, MO, 2005.

[21] N.R. Mead and E.D. Hough, "Security requirements engineering for software systems: Case studies in support of software engineering education," in *Proc. 19th Conf. Software Engineering Education and Training*. Los Alamitos, CA: IEEE Computer Society Press, 2006, pp. 149-158.

[22] S. Myagmar, A.J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirements," *Symposium on Requirements Engineering for Information Security (SREIS)*, 2005. http://www.projects.ncassr.org/multicast/papers/sreis05.pdf

[23] National Infrastructure Advisory Council (NIAC), "National strategy to secure cyberspace," U.S. Department of Homeland Security, Washington, DC, 2003.

[24] M. Newman, "Software errors cost U.S. economy $59.5 billion annually," National Inst. of Standards and Technology (NIST), Gaithersburg, MD, 2002.

[25] B. Palyagar, "Measuring and influencing requirements engineering process quality," in *Proc. AWRE 04, 9th Australian Workshop on Requirements Engineering*, 2004. Retrieved May 31, 2007 from http://awre2004.cis.unisa.edu.au/

[26] B. Palyagar, "A framework for validating process improvements in requirements engineering, in *RE 04-IEEE Joint International*, 2004, pp. 33-36.

[27] S. T. Redwine (Ed.), "Software assurance: A guide to the common body of knowledge to produce, acquire and sustain secure software, version 1.1." U.S. Department of Homeland Security, Washington, DC, 2006.

[28] T.L. Saaty, *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill, 1980.

[29] P. Saitta, B. Larcom, and M. Eddington, "Trike v.1 Methodology Document [Draft]," 2005.

[30] D. Shoemaker, N.R. Mead, A. Drommi, J. Bailey, and J. Ingalsbe, "SWABOK's fit to common curricular standard," in *Proc. 20th Conf. on Software Engineering Education and Training*. Los Alamitos, CA: IEEE Computer Society Press, 2007.

[31] G. Sindre and A. Opdahl, "Eliciting security requirements by misuse cases," in *Proc. TOOLS Pacific 2000*. Los Alamitos, CA: IEEE Computer Society Press, 2000, pp. 120-130.

[32] K. Soo Hoo, A.W. Sudbury, and A.R. Jaquith, "Tangible ROI through Secure Software Engineering," *Secure Business Quarterly* vol. 1, no. 2, Fourth Quarter 2001. http://www.sbq.com/sbq/rosi/sbq_rosi_software_engineering.pdf

[33] F. Swiderski, and W. Snyder, *Threat Modeling*, Microsoft Press, 2004.

[34] S. Butler, "Security Attribute Evaluation Method: A Cost-Benefit Approach," in *Proc. 24th Int. Conf. on Software Engineering*. Orlando, FL, May 19-25, 2002. New York, NY: ACM Press, 2002, pp. 232-240.

[35] S.L. Cornford, M.S. Feather, and K.A. Hicks, *DDP – A Tool for Life-Cycle Risk Management*, 2004. http://ddptool.jpl.nasa.gov/docs/f344d-slc.pdf

[36] Y.Y. Haimes, *Risk Modeling, Assessment, and Management,* 2nd ed. Hoboken, NJ: John Wiley and Sons, Inc., 2004.

[37] H.F.Lipson, N.R. Mead, and A.P. Moore, "A Risk-Management Approach to the Design of Survivable COTS-Based Systems," 2001. http://www.cert.org/research/isw/isw2001/papers/Lipson-29-08-a.pdf

[38] U.S. General Accounting Office. "Information Security Risk Assessment: Practices of Leading Organizations, A Supplement to GAO's May 1998 Executive Guide on Information Security Management," U.S. General Accounting Office, Washington, DC., 1999.

[39] G. Stoneburner, A. Goguen, and A. Feringa, "Risk Management Guide for Information Technology Systems,": National Institute of Standards and Technology, Gaithersburg, MD, Special Publication 800-30, 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[40] N.R. Mead, "Survivable Systems Analysis Method," 2002. http://www.cert.org/archive/html/analysis-method.html

[41] National Security Agency. "INFOSEC Assessment Methodology," 2004. http://www.iatrp.com/iam.cfm

[42] N.R. Mead, E. Hough, and T. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, CMU/SEI-2005-TR-009, ADA452453, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html.

[43] J. Allen, S. Barnum, R. Ellison, G. McGraw, and N.R. Mead, *Software Security Engineering: A Guide for Project Managers*. Upper Saddle River, NJ: Addison-Wesley, ISBN -13:978-0-321-50917-8, 2008.

[44] N.R. Mead, "Requirements Prioritization Case Study Using AHP," *Build Security In*, September 2006.

https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/requirements/534-BSI.html

[45] B.W. Boehm, *Software risk management*. Piscataway, NJ: IEEE Press, 1989.