An Evaluation of Cost-Benefit Using Security Requirements Prioritization

Nancy Mead
Travis Christian

June 2010

ABSTRACT: This article describes a comparison of six security requirements prioritization methods: analytical hierarchy process (AHP), accelerated requirements method (ARM) prioritization, priority poker, cost-benefit model, security investment decision dashboard (SIDD), and COCOMO-II security extensions.

OVERVIEW

When building complex systems, stakeholders must often prioritize requirements as part of the requirements engineering process. The development team may not implement all requirements due to lack of time, lack of resources, or changing or unclear project goals. In these cases it is important to define which requirements should be given priority over others.

Security Quality Requirements Engineering (SQUARE) is a requirements engineering process developed by the Software Engineering Institute's (SEI) CERT® Program at Carnegie Mellon University (CMU). SQUARE is a nine-step process that delivers categorized, prioritized, and validated security requirements. SQUARE prioritizes requirements based on completed risk assessment and requirement categorization steps [Mead 2006a] [SQUARE 2010].

Various prioritization techniques may be and have been used in SQUARE step 8. Though the official SQUARE tool uses the Analytical Hierarchy Process (AHP), SQUARE does not prescribe any specific method. This article describes a comparison of six prioritization methods using a previous SQUARE case study, which described the Accelerated Requirements Method (ARM) and Analytical Hierarchy Process (AHP) method, for context [Hough 2006]. The case study applied SQUARE to three large-scale software applications with security requirements. One of the projects, the basis for the current evaluation, provides nine security requirements and a risk assessment, the inputs to SQUARE prioritization. For reference, Table 1 shows the security requirements.

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

Table 1. Security requirements

SR-1	The system shall implement access control via a secure login screen.
SR-2	The system shall identify and authenticate all users who attempt to access it.
SR-3	The server-side components and files contained therein shall have their access restricted to authorized personnel.
SR-4	Fault tolerance shall be provided for the asset management system's essential services (IIS server, GIS server, and network lines).
SR-5	The system shall maintain data integrity via logged modifications and user access control.
SR-6	An access control system shall be configured for optimal information gathering for auditing purposes (access log and application log).
SR-7	The system shall recover from attacks, failures, and accidents in less than one minute.
SR-8	A backup shall consist of a complete reproduction of every file on the server.
SR-9	The system shall be able to provide full functionality from backup.

Each of the six prioritization methods was performed on this same set of requirements. For the methods involving the collaboration of multiple stakeholders, students familiar with software engineering and basic security concepts played the role of stakeholders in a software project that had yielded the given requirements. They ranked the requirements according to their own judgment, by whatever criteria the method called for.

METHODS EVALUATED

1. Analytical Hierarchy Process (AHP)
2. Accelerated Requirements Method (ARM) Prioritization
3. Priority Poker
4. Cost-Benefit Model
5. Security Investment Decision Dashboard (SIDD)
6. COCOMO II Security Extensions

AHP

The Analytical Hierarchy Process (AHP) is a general decision-making method for situations involving multiple decision factors. It uses pair-wise comparison to estimate the relative values of each pair of options and the consistency of the responses. By breaking the problem down into a comprehensive set of individual comparisons, AHP provides a consistent model for prioritizing many elements [Mead 2006b].

The existing SQUARE support tool implements AHP to prioritize security requirements [Tool 2010]. For each pair of requirements, users indicate their relative cost and relative value on 5-point scales. After each pair of requirements is compared, the tool calculates the average score for each requirement as a composite of both factors and prioritizes them accordingly. In the SQUARE implementation, the lead requirements engineer may adjust the final rank after discussing the results with the team.

AHP has a number of advantages that made it an excellent choice for implementation in the SQUARE tool. The method has already been proven effective in a previous case study as well as in the wider context of industry use. It splits the problem into specific criteria, in this case cost and value, that can be measured independently. While AHP does not specifically require multiple participants, it scales to any number, averaging their results into a final ranking.

The method's disadvantages are minimal and can be easily mitigated. AHP relies heavily on calculations for pair-wise comparison, but with tool support this is not an issue. However, since the number of comparisons grows quickly as the number of requirements increases, it may be unwieldy for a complex project. AHP's results may be partially subjective because they rely on relative rather than absolute scoring, but relative scores could be determined with some formality to obtain more objective results. It is worth noting that the mathematical model behind pair-wise comparison can also reflect how consistently a requirement has been scored across all of its pairings in relation to the other requirements.

For this study, participants performed AHP using the existing SQUARE tool [Tool 2010]. For each pair of requirements, the tool presents users with two 5-point scales, one to indicate which requirement is more valuable and one to indicate which is more costly. Each participant, playing the role of a stakeholder for a large project, received a SQUARE project at the prioritization step and completed the pair-wise comparison. For the set of nine requirements, each participant compared a total of 36 pairs. The tool automatically calculated and averaged the results shown in Table 2.

Table 2. AHP results

Rank	Requirement
1	SR-2
2	SR-3
3	SR-1
4	SR-5
5	SR-8
6	SR-4
7	SR-7
8	SR-9
9	SR-6

ARM Prioritization

The Accelerated Requirements Method (ARM) is a full requirements elicitation technique. ARM uses a simple prioritization activity that can be performed independently of ARM's other activities. Each requirement's priority is ranked directly, not broken down into multiple factors. ARM users divide the set of requirements equally among three levels of importance: low, medium, and high. These rankings are translated into 1, 3, or 9 points, respectively. The requirements are then prioritized according to the average number of points assigned to them by the users [Hough 2006].

This technique is very simple and requires no special knowledge or tool support. It is also inherently democratic. However, it also has the potential to be very subjective, as there are no explicit criteria on which to base the rankings. Stakeholders do not have to make specific comparisons between individual requirements, only categorize them. Because there are only three possible scores for each requirement, it is easy to have ties in the results, especially with a small number of users.

Participants, acting as stakeholders, received the set of security requirements and divided them equally among the high-, medium-, and low-priority categories. After the results were converted to numbers and averaged, ARM yielded the results shown in Table 3.

Table 3. ARM prioritization results

Rank	Requirement
1	SR-3
2	SR-2
3	SR-5
4	SR-9
5	SR-8
6	SR-1
7	SR-4
8	SR-6
9	SR-7

Priority Poker

Priority Poker is an experimental method derived from Planning Poker, the collaborative estimation game from Extreme Programming [Mountain Goat Software]. In Planning Poker, players use a deck of numbered cards to “vote” on the effort required to implement a user story. For each story, players (various stakeholders) reveal their estimates all at once, discuss the results, and repeat until a consensus is reached. Planning Poker seeks to minimize the influence of any one person’s opinion and involve all participants equally through discussion and voting.

Priority Poker uses the same rules as Planning Poker, except that requirements take the place of user stories, and players rank the overall criticality of a requirement rather than the effort needed to implement it. Like its inspiration, Priority Poker requires nothing more than a set of cards. The game can be played exactly as Planning Poker, but since priority is not necessarily a measurable value it can be confusing to rank the requirements by powers of 2. For Priority Poker, it would be more sensible to use descriptive priority levels than numbers: “Nice to have,” “Important,” “Very important,” “Critical,” and so on. Otherwise, the game can be played using a Planning Poker application or card deck. Overall it is similar to the ARM approach, except that players make their decisions iteratively and collaboratively.

Priority Poker is inherently democratic and collaborative. It encourages discussion and agreement among stakeholders. By revealing their estimates all at once, participants are less likely to defer their opinions to the first or most influential person to speak. However, the game remains highly subjective, as it relies on the gut feelings of each player to arrive at a common result. It is also easy to have ties among multiple requirements (which is not a problem for planning but is undesirable for prioritizing).

Priority Poker was played using the free web-based Planning Poker application [Mountain Goat Software]. As the game moderator introduced each requirement as a “story,” players voted using numbered cards until a consensus was reached. On reflection, using such an arbitrary number scale made it both more difficult to choose a starting point and easier to come to agreement. Players were very willing to change their votes because there was no clear definition of any particular number. It was not until after a few requirements had been prioritized that the numbers started to have much meaning. Thus, votes for the first few requirements were arbitrary because there was no context in which to compare one requirement or its priority to another. After a few rounds, the earlier results served as reference points to which votes could be compared. Using named priority levels rather than numbers would have made the decision-making process more concrete and given players some reference points during the early rounds. Nonetheless, even with arbitrary numbers the players were able to arrive at a meaningful result in the sense that requirements were prioritized in relation to each other rather than on an absolute scale. Table 4 shows the results of the game.

Table 4. Priority poker results

Rank	Requirement
1	SR-2
2	SR-3
3	SR-1
4	SR-5
5	SR-8
6	SR-4
7	SR-9
8	SR-6
9	SR-7

Cost-Benefit Model

Part of the SQUARE project developed a cost-benefit analysis framework to help small companies estimate the net value of information security improvements. The framework uses an economic model of the risks associated with information security to find the likely net value of an investment over time. Total costs and benefits of alternative investments are calculated based on their mitigation of categories of threats. Security investments are valued according to their ability to mitigate risk in various categories and thus reduce expected costs of those risks compared to the cost of implementation [Xie 2004].

The full scope of this framework is not necessary for prioritizing preselected requirements, but the risk model is very useful for estimating their net value. The model can be applied in a few different ways. For this evaluation, requirements were organized as categories of prevention by their ability to combat a specific security threat. The net value of a category then became the economic value of a requirement. This simplified the benefits of each requirement to a single reduction in risk.

For situations in which detailed data on security risks is available, requirements could be valued according to their ability to mitigate multiple risks across different categories. In this case, each requirement's total value would be represented by a total system value because each requirement would present a combination of categories of prevention at a specified cost—the means by which total system value is determined. In another approach, the model could be used to evaluate different combinations of requirements to determine the optimal set according to total system value. The actual use of the model can vary based on the available data, but the basic premise remains the same. Requirements represent the potential to provide economic value by reducing costs associated with security threats.

In this case there was no background data available, so risks and costs were reduced to simple estimates. The reduction of some relevant type of risk was estimated for each requirement, which served as a category of prevention. This risk reduction represented the economic value of implementing the requirement. Then the cost of implementation was estimated as an arbitrary cost factor on a relative scale of 1 to 10. The ratio of the value to the cost provided the net value of each requirement. Requirements were then ranked by net value. Table 5 shows the results.

Table 5. Cost-benefit model results

Rank	Requirement
1	SR-2
2	SR-3
3	SR-1
4	SR-5
5	SR-4
6	SR-8
7	SR-6
8	SR-9
9	SR-7

SIDD

The Security Investment Decision Dashboard (SIDD) is a framework developed by CERT for ranking the importance of an investment by certain weighted criteria. The base model includes seven criteria with 33 associated indicators. Stakeholders first rank the seven criteria in order of importance to the project, then the 33 indicators. Then stakeholders score each criterion and indicator from 1 to 5 on a question relating to the investment. The scores are weighted according to rank and used to produce an overall score for each investment [Allen 2008].

SIDD provides customizable, granular analysis of each option, using cost, benefits, and other factors both tangible and intangible. It requires some degree of formality and a fair amount of contextual data, but not necessarily hard numbers. Because SIDD is a CERT-proprietary framework, the actual formulas for calculating weights and final scores are not publicly available.

Some indicators are difficult to apply at the level of individual requirements, but SIDD is designed to be modular. Criteria and indicators can be added, modified, or removed as necessary. Since the numerical scores are all relative to each other rather than absolute measurements, indicators can simply be ignored as long as they are consistently ignored.

For this evaluation, several indicators that did not directly apply to implementing requirements were discarded across the board. This resulted in an effective score of 0 in those indicators for all requirements. Since relevant background data was not available from the earlier case study, the default criteria ranking was used. Participants scored indicators based on estimates and assumptions about the nature of the project and its business context. In a real project, this would be a much more formal process, using business policies to rank criteria and quantifiable metrics to determine scores. Table 6 shows the results of SIDD.

Table 6. SIDD results

Rank	Requirement
1	SR-2
2	SR-5
3	SR-3
4	SR-1
5	SR-8
6	SR-6
7	SR-9
8	SR-4
9	SR-7

COCOMO II Security Extensions

COCOMO (Constructive Cost Model) II is a popular cost estimation model for software construction developed by Barry Boehm in 1991 as a revision of the original COCOMO. COCOMO estimates the effort needed to produce a piece of software by breaking it down into a hierarchy of tasks and then finding the approximate size in lines of code (LOC) and the time (and therefore cost) to implement each part, thus estimating the whole project [Mead 2009]. There have been various extensions to COCOMO II, including an effort by the University of Southern California to include estimation for security implementation. This project, called COCOMO Security Extension (COSECMO), aimed to accurately predict the effort needed to implement security measures in software [Colbert 2006]. However, the project was terminated prior to full development, so we were not able to include it in our comparison.

COCOMO itself is widely accepted in the software engineering community, and there are various extensions and optimizations in use. It provides solid, traceable estimates for implementation costs based on detailed software and process specifications. However, its complexity makes it impractical without tool support. A security model with tool support would be useful for estimating the cost of security requirements before using another, more detailed method like SIDD. COCOMO considers only the cost of implementation, which should be paired with other factors to provide a better representation of priority.

As noted above, the security requirements could not be evaluated by this method because it was not fully implemented. Nonetheless, a similar model could offer potential for formal analysis of the cost of implementing security requirements. The results of such a technique could be used as reliable input to other methods such as those described earlier.

RESULTS

The prioritization methods fall into two general categories: democratic and metrics-based rankings. AHP, ARM, and Priority Poker are democratic. They are easy to perform but can be subjective. All stakeholders can participate regardless of role or expertise. SIDD and the cost-benefit model are metrics-based. They require considerably more effort and information to complete, but they are based on hard data. For smaller, less critical projects, or when security and cost data are unavailable, the democratic methods may be useful. When detailed data is available, the metrics-based methods provide a more stable and scientific approach that may be necessary for larger, complex systems.

Interestingly, the evaluated methods produced fairly similar results in this evaluation even though some of the initial input values were arbitrary and the mock stakeholders did not have the background information that real system stakeholders would possess. SR-2 and SR-3 were consistently ranked highest priority; SR-7 was almost unanimously ranked lowest. Across all methods, no single requirement varied in rank by more than three places. Table 7 shows a composite chart of all results.

Table 7. Results of all methods

	Requirement				
Rank	AHP	ARM	Priority Poker	Cost-Benefit	SIDD
1	SR-2	SR-3	SR-2	SR-2	SR-2
2	SR-3	SR-2	SR-3	SR-3	SR-5
3	SR-1	SR-5	SR-1	SR-1	SR-3
4	SR-5	SR-9	SR-5	SR-5	SR-1
5	SR-8	SR-8	SR-8	SR-4	SR-8
6	SR-4	SR-1	SR-4	SR-8	SR-6
7	SR-7	SR-4	SR-9	SR-6	SR-9
8	SR-9	SR-6	SR-6	SR-9	SR-4
9	SR-6	SR-7	SR-7	SR-7	SR-7

CONCLUSIONS

Each of the five performed methods proved applicable and useful. As noted above, a method such as the COCOMO-II Security Extensions, fully developed and with tool support, could also be useful for providing a cost estimate for SIDD or the cost-benefit model. Both SIDD and the cost-benefit model seem useful, but without real-world data it is difficult to determine their true accuracy. Even so, their results were fairly consistent with the other methods in this evaluation.

Overall, SIDD seems to be the most comprehensive method for prioritizing security requirements if the applicable information is available. Its primary drawback is that the spreadsheets (and thus the formulas) for calculating scores are for CMU/CERT use only. Lack of tool support currently makes each of the metrics-based methods less attractive options.

Of the methods that do not require hard data, AHP has proven most useful. It captures both cost and value estimates with relatively little effort, and it allows all stakeholders to participate. Also, the pair-wise comparison technique is already widely accepted and supported for similar activities.

NEXT STEPS

As previously mentioned, input to SIDD and the cost-benefit framework was based on proxy estimates and educated guesses rather than real-world data, but the results were nonetheless consistent. Both would be good candidates for application in a further case study with real-world data.

The methods presented here are those that seem to be the most applicable to prioritizing multiple parallel investments at the system requirement level. There are various other prioritization methods that may prove useful in this context, including Theory-W, Numerical Assignment, 100-Point Method, and industry standard cost-benefit valuation models [SQUARE 2010].

REFERENCES

[Allen 2008]

Allen, Julia H. *Making* Business-*Based Security Investment Decisions – A Dashboard Approach*. (2008).

[Colbert 2006]

Colbert, Ed; Wu, Dan; Chen, Yue; & Boehm, Barry. Cost Estimation for Secure Software & Systems. Los Angeles, CA: Center for Software Engineering, University of Southern California, 2006. http://csse.usc.edu/csse/TECHRPTS/2006/usccse2006-600/usccse2006-600.pdf

[Hough 2006]

Hough, Eric; Ojoko-Adams, Don; Chung, Lydia; & Hung, Frank. Security Quality Requirements Engineering (SQUARE): Case Study Phase III (CMU/SEI-2006-SR-003). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/library/abstracts/reports/06sr003.cfm

[Mead 2006a]

Mead, Nancy R. SQUARE Process. http://buildsecurityin.us-cert.gov/articles/best-practices/requirements-engineering/square-process (2006).

[Mead 2006b]

Mead, Nancy R. Requirements Prioritization Case Study Using AHP. http://buildsecurityin.us-cert.gov/articles/best-practices/requirements-engineering/requirements-prioritization-case-study-using-ahp (2006).

[Mead 2009]

Mead, Nancy R.; Allen, Julia H.; Conklin, W. Arthur; Drommi, Antonio; Harrison, John; Ingalsbe, Jeff; Rainey, James; & Shoemaker, Dan. Making the Business Case for Software Assurance (CMU/SEI-2009-SR-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09sr001.cfm

[Mountain Goat Software]

Mountain Goat Software. Planning Poker. http://www.planningpoker.com/

[SQUARE 2010]

SQUARE Instructional Materials http://www.cert.org/sse/square/square-tool.html (2010).

[Tool 2010]

SQUARE Tool. http://www.cert.org/sse/square/square-tool.html (2010).

[Xie 2004]

Xie, Nick, & Mead, Nancy R. 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