Requirements Elicitation Case Studies Using IBIS, JAD, and ARM

Nancy Mead

September 2006

ABSTRACT: This article describes a tradeoff analysis that can be done to select a suitable requirements elicitation method and the results of trying three methods in some case studies. It is a companion to the requirements elicitation introduction.

ACKNOWLEDGEMENT: This material is extracted and adapted from a more extensive case study report by Carnegie Mellon graduate students Lydia Chung, Frank Hung, Eric Hough, and Don Ojoko-Adams [Chung 06].

BACKGROUND

The case studies were conducted by a team of Carnegie Mellon graduate students under my supervision during a full-time semester-long project [Chung 06].

The students selected and applied several elicitation methods in the course of developing security requirements for three clients.

The student team was constrained in the amount of time to spend on the process, since they were managing the schedule for a one-semester project. It was also the case, however, that the clients had limited time to devote to this exercise. Therefore, you will see time constraints mentioned several times in the case study. This refers to the combination of student and client time constraints.

While results may vary from one organization to another, the discussion of how we applied the various methods should be of general use.

SELECTING AN ELICITATION METHOD

The following is a partial list of elicitation methods that could be considered for eliciting security requirements. There are, of course, other methods, but this list provides a good sample. The elicitation methods can be ranked using a tabular form (see Table 1). In this article, we have filled in the values provided by the student team for the various methods. Each method was rated according to the desired features, and the rankings were simply added to provide a summary re-

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

sult. A weighted average could also have been used if some features were considered to be more important than others. Note that this sort of evaluation is subjective, particularly since the students were working under time constraints and did not have prior experience with this, so results may vary from one organization to another:

- misuse cases [Sindre 00, McGraw 06]
- Soft Systems Methodology (SSM) [Checkland 90]
- Quality Function Deployment (QFD) [QFD 05]
- Controlled Requirements Expression (CORE) [Christel 92, SDS 85]
- issue-based information systems (IBIS) [Kunz 70]
- Joint Application Development (JAD) [Wood 95]
- feature-oriented domain analysis (FODA) [Kang 90]
- critical discourse analysis (CDA) [Schiffrin 94]
- Accelerated Requirements Method (ARM) [Hubbard 00]

Evaluation Criteria

In our case studies, in order to evaluate the elicitation methods, the following criteria were used. For this case study, a comparison matrix of desirable features was developed by the student team. We recommend that each organization develop its own matrix of desirable features as part of its elicitation method selection process. The following are example criteria that may be useful in selecting an elicitation method, but certainly there are other criteria that you could use. The main point is to use criteria and to have a common understanding of what they mean.

adaptability: The method can be used to generate requirements in multiple environments. For example, the elicitation method works equally as well with a software product 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 defines a CASE tool as "a computer-based product aimed at supporting one or more software engineering activities within a software development process" [SEI 06].)

- stakeholder acceptance: The stakeholders are likely to agree to the elicitation method in analyzing their requirements. For example, the method isn't too invasive in a business environment.
- easy implementation: The elicitation method isn't 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 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.

Note that this approach presumes that all criteria are equally important. If some criteria are more important than others, a weighted average can be used. 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. The elicitation methods can then be ranked using a tabular form, such as the example shown in Table 1.

In Table 1, the individual scores reflect the students' assessment of how well that particular method satisfied each criterion. For example, IBIS was assessed to have a good supporting CASE tool, so it is ranked 3 in that category, whereas ARM has no CASE tool, so it is ranked 1 in that category. The total score is simply a summary of the individual scores. This is not intended to be an actual recommendation to use a specific method. You can develop your own comparison criteria and ratings.

Table 1. Comparison of elicitation methods

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

	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

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 the time and effort constraints that we were working with. We considered not just the total score. Implicitly, the learning curve was an important factor, since the students were constrained to complete the work in a single semester. Also, the team attempted to select methods that were not too similar to one another, so as to have some variety.

Additional considerations

It is possible that a combination of methods may work best. You should be consider this as part of the evaluation process, assuming that you have sufficient time and resource to assess how methods may be combined and to actually combine them. You should also consider the time necessary to implement an elicitation method and the time needed to learn a new tool that supports a method. Selecting an elicitation method that meets the needs of a diverse group of stakeholders aids in addressing a broader range of security requirements.

SECURITY REQUIREMENTS ELICITATION CASE STUDIES

IBIS, ARM, and JAD were used to elicit security requirements for projects Alpha, Beta, and Delta, respectively. In this section, we describe the Carnegie Mellon team's experience in the application of each method, providing recommendations when possible.

IBIS

IBIS was developed in the 1970s to improve the definition, discussion, and resolution of "wicked" problems. That is, the method is intended to address issues that are ill defined or hotly contended among stakeholders.

In IBIS, all problems are decomposed into issues, which are phrased in the form of open questions to the stakeholders. One question might be, for instance, How

should the system guard against insider threats, if at all? Each issue is then resolved by proposed positions, which are resolutions to the issue put forth by the stakeholders. Every position has corresponding arguments that either support or oppose the position.

The requirements engineer is tasked with recording the articulation of issues, positions, and arguments. The results are then presented in the form of an IBIS map (IM). Figure 1 is an example of such a map. The IBIS maps are analyzed by the requirements engineering team and stakeholders to elicit the actual security requirements.

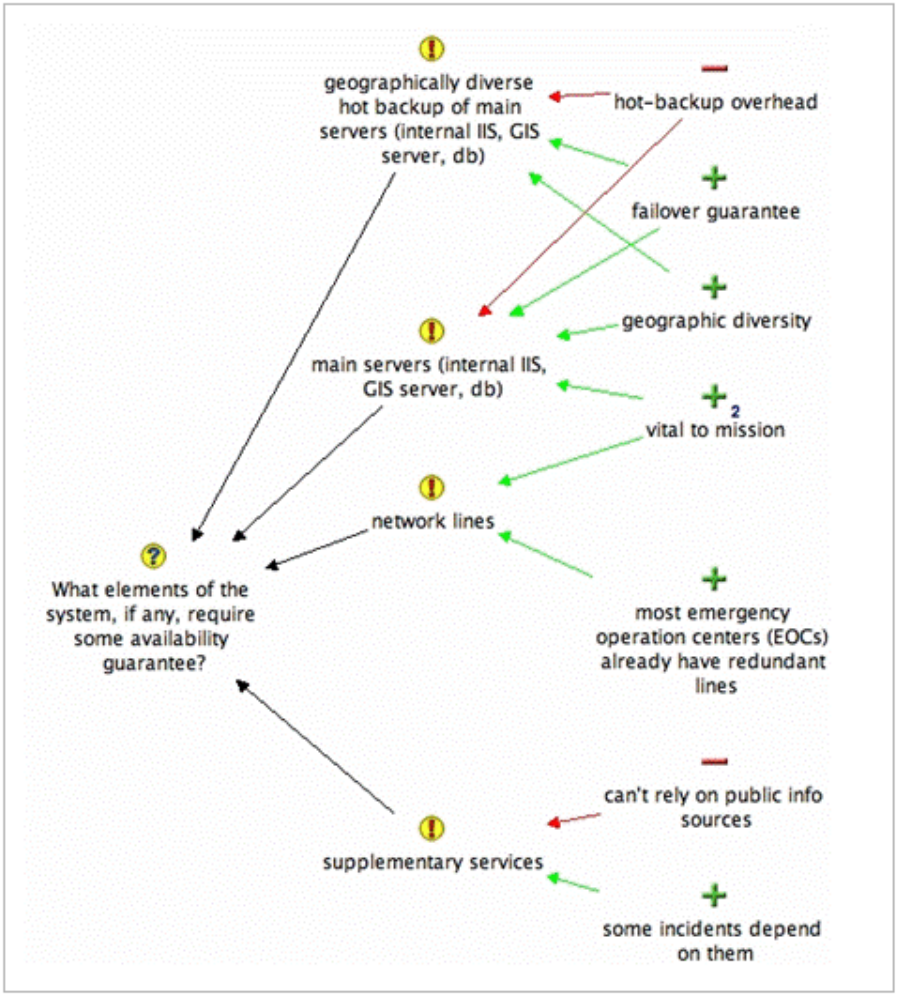

Figure 1. Example of an IBIS map generated with Compendium

Application of IBIS in the Case Study

First, the team formulated a set of questions that would likely cover every aspect of security that could affect the system. We tried to identify any and all questions that would cover confidentiality, availability, and integrity of the system. A representative sample of these questions can be found below in Table 2.

Table 2. Example IBIS questions

No.	Example IBIS Questions
1	What level of training, if any, will users of the system require?
2	What elements of the system, if any, require some level of confidentiality?
3	What data elements, if any, require some level of integrity guarantee?
4	How should the system implement data integrity guarantees?
5	What elements of the system, if any, require some availability guarantee?
6	What type access control, if any, should the system utilize?
7	What measures, if any, should the system utilize to protect against insider threats?
8	While this question worked well for a mature system undergoing development, it's not really relevant to a product undergoing design.
9	What secure coding techniques, if any, are necessary?
10	How should the system's essential services be protected against threats?
11	What type of intrusion-detection measures should be in place, if any?
12	How should the system record intrusions and intrusion responses, if at all?

We then created a set of IBIS maps using the Compendium CASE tool, which is freely available from the Compendium Institute's Web site [CI 08]. This software was very easy to use, and we were able to create our set of maps quickly.

Since the maps didn't contain any inherent identification information, we attached a table to each map to identify and reference it. The tables, which we call

IBIS map descriptors, contain information such as the positions, security attributes that are affected, and related security goals. Figure 2 shows one of our IBIS map descriptors.

Number	IMD-04
Issue	What type of authentication, if any, should the system utilize?
Positions	Integrated Windows logon SSL-protected login screen
Security Attributes Affected	x Confidentiality __Integrity __Availability
Survivability Attributes Affected	x Resist __Evolve __Recognize __Recover
Related IBIS Map Descriptors	IMD-15, IMD-16
Related Security Requirement(s)	SR-01, SR-02, SR-03, SR-05, SR-06

Figure 2. Example IBIS map descriptor

Using these maps, along with feedback from the Alpha project stakeholders, we were able to generate an initial set of security requirements for the system. Unfortunately, IBIS does not provide a mechanism for translating the maps into requirements, so our requirements were based on feedback from Alpha during the meetings and our own recommendations. These appear below in Table 3.

Table 3. Categorized security requirements

Category	No.	Requirement
Resistance	SR-01	The system shall implement access control via a secure login screen.
	SR-02	The system shall identify and authenticate all the users who attempt to access the system.

	SR-03	The server-side components and files contained therein shall have their access restricted to authorized personnel.
	SR-04	Fault tolerance shall be provided for the 1st Responder Management System's essential services (IIS server, GIS server, and network lines).
Recognition	SR-05	The system shall maintain data integrity via logged modifications and user access control.
	SR-06	An access control system shall be configured for optimal information gathering for auditing purposes (access log and application log).
Recovery	SR-07	The system shall recover from attacks, failures, and accidents in less than one minute.
	SR-08	A backup shall consist of a complete reproduction of every file on the server.
	SR-09	The system shall be able to provide full functionality from backup.

Recommendations

Choose Interview Questions Wisely

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. As is usually the case in requirements elicitation, 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. The IBIS interview, in this case, simply needs to record the statements made during the discussion.

Questions that utilize the word should are well suited for IBIS interviews, since they force the stakeholder to provide answers that aid in eliciting the actual security requirements. For instance, What type of intrusion detection mechanisms, if any, should the system utilize? The IBIS question should be less concerned with the implementation--that is, with how the requirement will be met. However, using the word how is appropriate in cases where there is no doubt among all stakeholders of a particular requirement. For example, How quickly should the system recover from failure? implies that all stakeholders agree that the system

must recover from failure. The IBIS interview must be certain, likely through artifact collection, that there is no ambiguity among stakeholders on this requirement.

We found that direct, nonleading questions work very well. For instance, we recommend asking questions such as What measures should be in place to protect against configuration errors, if any? rather than How should the system respond to configuration errors? The latter question implies an agreed-upon requirement, even though there may be debate as to whether the topic of the question is a requirement at all. Much like direct examination in judicial proceedings, direct questions allow the stakeholders to provide an honest, unbiased, and complete response to each question.

The IBIS questions must strike a balance between being generic enough for reuse in subsequent projects and too general to elicit specific responses. For example, the question What type of security mechanisms should be used? can be reused from project to project, but it is far too general to elicit a useful and complete response from the stakeholders. Should the system use the Snort intrusion detection system? is a question that would lead to a detailed response, but this question is overly specific. Instead, the IBIS interviewer should ask, What type of intrusion detection system, if any, should the system use?

The requirements engineer should also consider previous system artifacts. Upon examination of our first draft of IBIS questions for the stakeholders, we discovered that many of them would have been answered in the artifact collection phase of the process. Due to time constraints and our existing general knowledge of the stakeholders' project, we mistakenly included some artifact-related questions such as What are the minimum computing requirements to run the software? We suggest that IBIS questions be formulated carefully to exclude questions that have already been answered.

Include a Variety of Stakeholders

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. By presenting differing viewpoints, the stakeholders are forced to discuss and analyze one another's positions while, we hope, reaching a consensus. Examples of stakeholders include software developers, hardware developers, network engineers, security managers, executives, end users, and marketing and finance managers. During the setup phase, well ahead of the actual interview session, the requirements engineer should emphasize the need to include as many stakeholders as possible in the discussion.

Avoid Presenting Cluttered IBIS Maps

The Compendium software tool was easy for the student team to use and effective in generating IBIS maps. Note that the client did not have the opportunity to use the tool. Normally we would expect the requirements engineers to be using the tool, and this was the role the student team was filling. In order 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 team 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 we received from the stakeholders indicated that such a hierarchical map structure would have been more beneficial in handling some of the larger maps.

Organize the maps to reduce clutter and orient each map in a consistent manner when printing out the maps for stakeholder review. We found that the stakeholders had some difficulty in determining the correct orientation of the pages when reviewing the maps.

ARM

ARM is designed to elicit, categorize, and prioritize security requirements. The method is discussed in detail in Hubbard's doctoral dissertation and later summarized in a published paper [Hubbard 00]. The student team spent two weeks completing the ARM process with Project Beta.

The ARM method includes the three phases shown in Figure 3. We discuss each phase in detail.

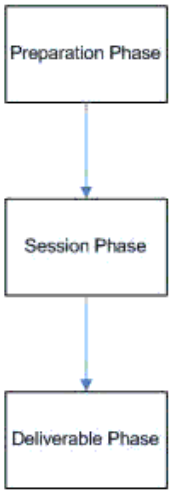

Figure 3. Three phases of ARM

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.

During the initial steps of the Preparation Phase, the team prepared a feedback form composed of seven questions for the stakeholders. Such questions and other written interactions are a standard part of the ARM method:

1. What are the goals of this project?
   The goal statement usually describes the purpose of the project in one sentence.
2. What are the objectives of this project?
   The objective statement is derived from the goal and can be treated as a goal with a detailed statement. A project can typically have five to seven objectives.
3. What are the PSC for this project?
   PSC are used to describe what factors can help the project become a successful project. PSC can be both business and functional criteria.
4. What is the scope for this project?
   In-scope items are topics that are suitable to discuss in the Session Phase. Out-of-scope items are topics that are unsuitable to discuss in the Session Phase.
5. What are the partitions of this project?
   According to the goal, objective, PSC, and scope, a partition breaks the project into small pieces that are correlated with one another and highly cohesive.
6. Who are the participants?

According to the goal, objective, PSC, scope, and partition, the participants are those who are suitable to join the meeting.

7. What are the environmental aspects?
   To have a successful meeting, the team should prepare environmental and logistical arrangements that include room selection, technology arrangements, and refreshments.

The stakeholders spent two business days to complete and return the form. While the stakeholders worked on the form, the team prepared a memorandum containing goals, objectives, PSC, preliminary scope, partition definitions, participants, and logistic arrangements. Participants were asked to read the memorandum before the Session Phase in order 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.

After reviewing the stakeholders' feedback forms, the team decided that there were no outstanding questions, issues, or confusion with the process and that another meeting with the stakeholders was not necessary before beginning the Session Phase.

Session Phase

The Session Phase is the heart of the ARM process, and it entails the following 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, the team made logistical arrangements to ensure that the meeting would go smoothly. The following is a brief summary of the arrangements:

- Supply note cards for participants to write more sentences.
- Supply tape to attach cards to the wall.
- Provide the following items to aid team members:
  - detailed individual job assignments
  - security requirement form: The words written in the cards could have been too small to read, so the team prepared a form that could project the content of the cards on the screen.

  - grouping form: In the Organize step of the process (Step 3), the team could show the categorized result on the screen immediately.
  - details form: In the Detail step (Step 4), the team could fill in the outputs from the participants on the form.
- Supply package for each participant, including
  - memorandum
  - prioritization form
  - feedback form
  - Session Phase slides
  - Preparation Phase slides
  - scratch paper
  - note cards (four)

Executive Sponsor Commentary

Due to time constraints, the team decided to omit this step. Also, the team understood that the participants already possessed the information, which would have been conveyed during the Preparation Phase. However, the team did provide a brief introduction to ARM and the procedures of the Session Phase meeting.

Scope Closure

The team also decided to omit this step because its primary purpose is to prepare the participants for the following steps. However, since the participants worked closely together in the same department, they required little preparation time to familiarize themselves with security issues.

BON: Brainstorm, Organize, and Name

The BON step provided an efficient way to elicit the candidate requirements from participants. First, the team asked the participants the focus question, which was crafted to tie to the goals, objectives, and scope of the project. In this case, we chose the following focus question:

An important security requirement of the Beta Application is __

Based on their professional experience and security knowledge, the participants were asked to write down seven important security requirements on scratch paper within the time limit of seven minutes.

Afterwards, the team asked the participants to write down their top three or four security requirements on cards within three minutes. The team then collected the cards and put the candidate security requirements on the wall. The 24 candidate security requirements produced are listed below in Table 4.

Table 4. Initial requirements produced in ARM

1	The ability to securely transmit data to remote sources	13	Accountability (who did what, when, how...)
2	The preservation of data integrity	14	Integrity (assurance in data protection and validity)
3	The enforcement and usability of an access control system	15	Indelibility (deletions and retractions are logged)
4	Security must be manageable and not hinder business (where possible).	16	Integrity
5	There must be a strong, reliable authentication process.	17	Access control
6	Information must be kept private from the outside world.	18	Confidentiality (encryption, etc.)
7	Consistent application program interfaces (APIs)	19	Partitioned data store (public read only and private read/write)
8	Data integrity	20	Selectively secure communication with outside entities.
9	Authentication and access control	21	Represent and support segmented disclosure.
10	Strong authentication	22	Role-based, restricted view, edit, and action access (e.g., summary report information, public for particular people)
11	Reduce or eliminate risks of inappropriate behavior	23	Available 24/7 via remote authenticated access and secure
12	Granular access to data for users (operators) and customers	24	Key action audit (e.g., attribution of who pressed the publish button and from where, and what changes were made)

In the Organize step, all the participants reviewed the candidate security requirements generated during the brainstorming session to see whether any duplicate or inadequate security requirements were included. Then the participants thoroughly discussed what they thought were important requirements. This step

provided an opportunity for the participants to share their security concerns about the project. After a period of discussion and debate, they deleted seven candidate security requirements as redundant or inappropriate.

Specifically the participants removed requirements 1, 2, 5, 9, 14, 16, and 17. The remaining requirements are shown in Table 5.

Table 5. The remaining requirements after initial eliminations

3	The enforcement and usability of an access control system	15	Indelibility (deletions and retractions are logged)
4	Security must be manageable and not hinder business (where possible).	18	Confidentiality (encryption, etc.)
6	Information must be kept private from the outside world	19	Partitioned data store (public read only and private read/write)
7	Consistent APIs	20	Selectively secure communication with outside entities)
8	Data integrity	21	Represent and support segmented disclosure.
10	Strong authentication	22	Role-based, restricted view, edit, and action access (e.g., summary report info)
11	Reduce or eliminate risks of inappropriate behavior.	23	Available 24/7 via remote authenticated access and secure
12	Granular access to data for users (operators) and customers	24	Key action audit (e.g., attribution of who pressed the publish button and from where, and what changes were made)
13	Accountability (who did what, when, how...)		

In the Name step, the participants were instructed to group the selected security requirements and create names for each group. However, the participants instead engaged in a spirited discussion that combined grouping, naming, and categorizing. Thus, security requirements, groups, and names were generated together. In the end, the participants categorized security requirements into six groups, each containing one to four security requirements. Table 6 lists the groups and the requirements contained in each.

Table 6. Grouped requirements

Group A: Confidentiality	• Information must be kept private from the outside world (Requirement 6). • Selectively secure communication with outside entities (Requirement 20).
Group B: Access Control	• Role-based, restricted view, edit, and action access (e.g., summary report information, public for particular people) (Requirement 22) • The enforcement and usability of an access control system (Requirement 3) • Granular access to data for users (operators) and customers (Requirement 12) • Represent and support segmented disclosure (Requirement 21)
Group C: Data Integrity	• Partitioned data store (public read only and private read/write) (Requirement 19) • Indelibility (Requirement 15)
Group D: Manageability	• Accountability (Requirement 13) • Key action audit (e.g., attribution of who pressed the publish button and from where, and what changes were made) (Requirement 24) • Auditing capabilities (derived requirement)
Group E: Usability	• Security must be manageable and not hinder business (where possible) (Requirement 4). • Available 24/7 via remote authenticated access (Requirement 23) • Consistent APIs (Requirement 7) • Reduce or eliminate risks of inadvertent behavior (Requirement 11).
Group F: Authentication	• Strong authentication (Requirement 10)

Requirements 8, data integrity, and 18, confidentiality, were deleted in this step, and a fourth requirement was added to Group E.

Details: Benefits, Proof, Assumptions, Issues, and Action Items

In Step 4, the participants were 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 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?

Due to the time constraints of the work session, the team skipped questions 6, 8, 9, and 10. However, the participants found it difficult to ask the questions of each requirement in turn, both because it was tedious and because some of the requirements were interrelated. Instead, the participants reviewed all the security requirements together, not individually.

The participants took an extended period of time to review the questions. Each question prompted a series of discussions and generated significant feedback from the participants. In the course of their discussion, the participants reviewed, reworded, and redefined those incomplete, incorrect, or ambiguous security requirements. The participants also encouraged one another to discover the true security requirements, not just recommendations. These were their basic assumptions:

- All equipment exists and is physically secure.
- Selected external parties have read/write access.
- Base system security is installed, current, and active.
- Classified information exists on the system and must be protected.
- Different levels of access within the development team are supported.

Prioritization

In the BON step of ARM, the participants generated the candidate security requirements of their project. They further modified and redefined the projected security requirements in the Details step to ensure that the requirements were unambiguous, clear, and concise. Note that this particular prioritization approach was not used as a standalone approach in our case studies but only in the context of ARM as an elicitation method.

The Prioritization phase of the ARM process began with the team providing instructions to guide participants to label each requirement A, B, or C, where A stood for most important, B stood for very important, and C stood for important. The rankings had to be assigned equally across the security requirements. Partic-

ipants prioritized each security requirement based on their professional knowledge and the importance of the requirement to the project. In this particular case, the participants completed the phase in 10 minutes. After the session concluded, the team calculated the scores. First, the team substituted the rankings A, B, and C with numeric values 9, 3, and 1, respectively. Then, the team calculated the average score of each requirement. The results are shown in Figure 4.

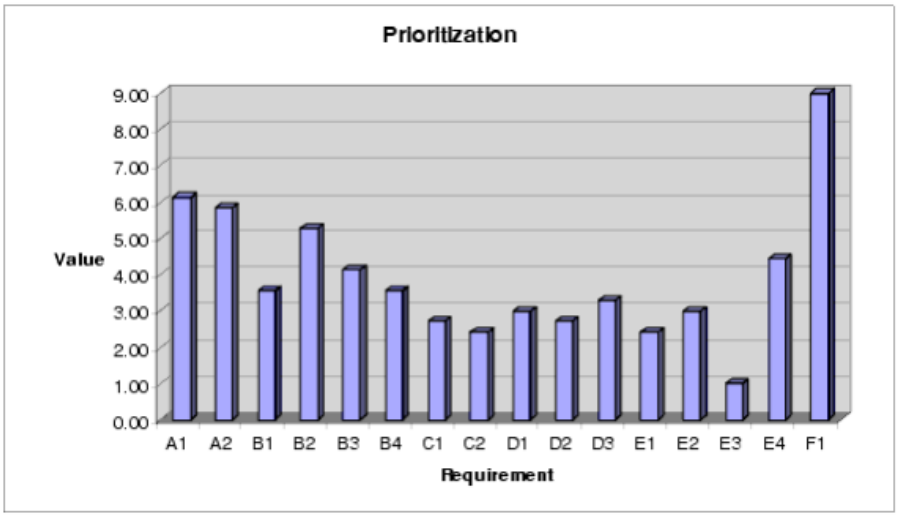

Figure 4. Ranked score of the requirements

Listed in order of priority and stated in a verifiable manner, these are the final requirements:

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 information 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, 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.

It is interesting to note that all participants agreed that Requirement 10, strong authentication, was a high priority and that Requirement 7, consistent APIs, was a low priority. Moreover, the participants thought the requirements in Group A, Confidentiality, and Group F, Authentication, were the most important ones, since all the requirements in those categories were ranked highest.

Based on the result of the prioritization, the participants could then plan to implement their security requirements. Therefore, they could use their limited resources effectively and maximize their satisfaction of the security of the application within application development time and budget constraints.

Stakeholders' Feedback

In the final portion of the Session Phase, the team requested that the participants fill out the feedback form that was used to collect useful information to improve the method. Through the feedback form, the team hoped to elicit the pros and cons of the Session Phase, areas for improvement, the experience of the participants during the meeting, and most importantly, how well the phase was able to elicit adequate security requirements. On the feedback form, three questions were asked:

- 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?

The participants mentioned that the meeting time was not long enough to permit them to generate detailed, useful security requirements. Also, some participants

thought that narrowing down to three security requirements might make those requirements too high level to be useful, and that more security requirements would be better for the project.

Many participants agreed that the Organize step was very important. In the Organize step, they generated and revised many ideas. Categorizing requirements allowed them to determine what was sensitive, critical, and less important. Additionally, organizing ideas seemed to promote a final demarcation of proposed requirements. The participants suggested that the session would have been more productive if more structure had been added to the categorization step and terms had been defined in advance.

The stakeholders' reaction to this phase was mixed. On one hand, most of the participants thought that the Session Phase promoted discussion and brought consensus regarding what they were trying to build. On the other hand, some of the participants regarded the process as less structured than they had anticipated.

Recommendations

Overall, ARM was extremely effective and, true to its name, a rapid method of collecting requirements. We found that by simply choosing the correct focus question, the process was very easily adapted to elicit security requirements. Note that the shortcuts taken may have biased the results.

In retrospect, loose time management was the greatest flaw in this process. 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.

Our results for the ARM method may be slightly biased since the participants were all security experts already. As such, they were able to easily generate a comprehensive set of security requirements. In future sessions, it's unlikely that all of the participants will have such a background; thus the requirements engineering team may need to review some security concepts with the participants before the session begins.

JAD

JAD brings users and technical professionals together to discover the functional requirements in software development. The centerpiece of JAD is a structured workshop known as the JAD Session, in which all stakeholders design a system or piece of software. JAD includes five distinct phases:

1. project definition
2. research

3. preparation
4. the JAD session
5. the final document

Project Definition Phase and Research Phase

The Project Definition Phase identifies the purpose, scope, objectives, assumptions, and open issues of the project by interviewing the managers from the users' departments.

The Research Phase focuses on collecting more detailed information about user requirements. In this phase, the work flow and preliminary specifications of data elements, screens, and reports are obtained by interviewing users.

The team decided to combine the Project Definition and Research Phases because only a few stakeholders could attend both meetings. The users of the project were intended to be the general public. Thus, our stakeholders could play both the manager and user roles.

In the first two phases of JAD, instead of interviewing the stakeholders, as intended by the process, the team prepared a list of questions for the stakeholders. The team did this because contact with the stakeholders was by teleconference, and the stakeholders had existing documents answering all the questions the team asked.

The stakeholders were asked the following questions:

- What is the purpose of the project?
- What is the scope of the project?
- What are the management objectives of the project?
- What are the security objectives of the project?
- What are the functions of the project?
- What are the constraints of the project?
- What are the assumptions of the project?
- What are the open issues of the project?
- Who are the participants in the project?
- What is the work flow of the project?

Preparation Phase

The team decided not to use any visual aids in the Preparation Phase, because the main stakeholder was in teleconference with the team.

The JAD Session Phase

The JAD Session Phase, the heart of JAD, addresses work flow, data elements, screens, reports, and open issues. Because JAD is designed mainly for functional (end user) requirements, there are some steps not suitable for discussing security requirements. Thus, the team decided not to go over the work flow, data elements, screens, and reports steps in the phase. Therefore, the only remaining step was to discuss open issues.

The stakeholders initially provided 16 open issues, but only 8 of them were security related. Based on the document from the first two steps, the team generated 11 additional open issues:

- How can you provide high availability? In the Web tier? In the database tier?
- What is your approach to version control?
- What is your approach to configuration management?
- What is your approach to defect and issue management?
- What is the testing method to be used in the proposed project?
- Do you maintain a separate environment for testing, or is testing performed on development servers?
- What is the software and system architecture?
- What are the security model and security configuration and implementation details of the software architecture?
- Why are you considering secure sockets layer (SSL) to authenticate users and provide privacy?
- What is the difference between HTTP and Remote Method Invocation (RMI) in regards to privacy?
- How should the integrity of the site be protected?
- How can unauthorized changes to the project affect the mission?
- What are the best procedures to guarantee these actions are recorded?
- Does the stipulation that all or most of the code be open source present any potential security issues?
- What are the differences between clustering and active or passive failover in regard to availability?
- How will you manage users and authorization?
- Is 100% uptime necessary for the project?
- Why don't you need version control to be built into the project?
- Why did you select the version-control system currently in use?

The team conducted an interview and generated the security requirements based on the answers provided from our stakeholders. Table 7 lists the requirements that were generated.

Table 7. Security requirements produced from the JAD session

SR1	The Web site shall provide reliable information to the users who have legitimate access to the Web site.
SR2	The Web site shall ensure that only authenticated users can access the protected content of the Web site.
SR3	The Web site shall protect the authenticated users' privacy by securing the communication channel.
SR4	The Web site shall ensure the integrity of content that is provided to the users by using authentication, authorization, and access control.
SR5	The Web site shall enable version control in both the content of the Web site and the development software.
SR6	The Web site shall enable auditing features that log all content modifications, work flow state transitions, access failures, and authentication attempts.
SR7	The Web site shall set up clustering to make the service sustainable when disaster occurs.

Recommendations

Since the work flow, data elements, screens, and reports steps of JAD were not suitable for discussing security requirements and were therefore excluded, the 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. The team thus 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.

The quality of the security requirements generated relied on the quality of the questions asked during the interview. This relationship posed a great risk for the JAD method. In this case, the team produced a number of different questions simply because the stakeholders at Delta were security professionals and had already considered the kinds of questions used in the Alpha and Beta cases. If

the stakeholders had been less security conscious, the results would have been very different from this case. The team may not have been able to obtain as much information and may not have grasped the core security issues of the project.

The variety of the participants in the JAD Session Phase is very important as well. In this case, there were only a few stakeholders participating in the phase, and there was no discussion between them. The team recommends that the JAD Session should involve all the stakeholders and that the facilitator should encourage them to share their opinions.

RESULTS SUMMARY FOR ALL THREE ELICITATION METHODS

ARM seemed better suited to elicitation of security requirements than either IBIS or JAD. The JAD results were similar to those that would be obtained by an unstructured interview process. JAD seemed more suited to end user functional requirements. There was 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. In order to get a good set of security requirements using IBIS, the requirements engineering team would have to carefully generate targeted interview questions. Detailed results for JAD and IBIS can be found in the case study report [Chung 06]. All three elicitation methods had the weakness that they were originally designed to focus on features. As a consequence, they tended to focus on security features (e.g. encryption, password utilization) and therefore did not consider security as a system property.

STATUS AND FUTURE PLANS

These case studies are part of the Security Quality Requirements Engineering (SQUARE) project [Mead 05]. Since these case studies were completed, we have published a report on how to compare SQUARE with other security requirements engineering methods [Mead 07]. We have also published a report examining ways of integrating SQUARE with popular life cycle models [Mead 08]. We have developed a prototype tool. We have also developed educational materials that can be downloaded.

We currently plan to extend SQUARE for acquisition and to develop a robust tool that provides both analysis and documentation support.

REFERENCES

[Christel 92]

Christel, M. & Kang, K. Issues in Requirements Elicitation (CMU/SEI-92-TR-012, ADA258932). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992.

[Checkland 90]

Checkland, P. Soft System Methodology in Action. Toronto, Ontario, Canada: John Wiley & Sons, 1990.

[Chung 06]

Chung, L.; Hung, F.; Hough, E.;Ojoko-Adams, D. Security Quality Requirements Engineering (SQUARE): Case Study Phase III (CMU/SEI-2006-SR-003). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

[CI 08]

The Compendium Institute, 2008.

[Hubbard 00]

Hubbard, R.; Mead, N.; & Schroeder, C. "An Assessment of the Relative Efficiency of a Facilitator-Driven Requirements Collection Process With Respect to the Conventional Interview Method." Proceedings of the International Conference on Requirements Engineering. June 2000. Los Alamitos, CA: IEEE Computer Society Press, 2000.

[Kang 90]

Kang, K. C.; Cohen, S. G.; Hess, J. A.; Novack, W. E.; & Peterson, A.S. Feature-Oriented Domain Analysis Feasibility Study (CMU/SEI-90-TR-021, ADA235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[Kunz 70]

Kunz, W. & Rittel, H. "Issues as Elements of Information Systems," Working Paper 131. Berkeley: Institute of Urban & Regional Development, University of California, 1970.

[McGraw 06]

McGraw, G. Software Security: Building Security In, Boston, MA: Addison-Wesley, 2006, pp. 205-222.

[Mead 05]

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.

[Mead 07]

Mead, N. R. How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods (CMU/SEI-2007-TN-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, August, 2007.

[Mead 08]

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). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, May, 2008.

[QFD 05]

QFD Institute. Frequently Asked Questions About QFD (2005).

[Schiffrin 94]

Schiffrin, D. Approaches to Discourse. Oxford, England: Blackwell Publishers Ltd, 1994.

[SDS 85]

Systems Designers Scientific. CORE--The Method: User Manual. London, England: SD-Scicon, 1986.

[SEI 06]

Software Engineering Institute. What is a CASE Environment? (2004).

[Sindre 00]

Sindre, G. & Opdahl, A. L. "Eliciting Security Requirements by Misuse Cases," 120-131. Proceedings of the 37th International Conference on Technology of Object-Oriented Languages (Tools 37-Pacific 2000). Sydney, Australia, November 20-23, 2000. Los Alamitos, CA: IEEE Computer Society, 2000.

[Wood 95]

Wood, J. & Silver, D. Joint Application Development, 2nd ed. New York: Wiley, 1995.

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM-0001120