Reinforcing Security Requirements with Multifactor Quality Measurement

Hanan Hibshi[1,2], and Travis D. Breaux[1]
Institute for Software Research, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA[1]
College of Computing, King Abdul-Aziz University, Jeddah, Saudi Arabia[2]
{hhibshi, breaux}@cs.cmu.edu

Abstract— Choosing how to write natural language scenarios is challenging, because stakeholders may over-generalize their descriptions or overlook or be unaware of alternate scenarios. In security, for example, this can result in weak security constraints that are too general, or missing constraints. Another challenge is that analysts are unclear on where to stop generating new scenarios. In this paper, we introduce the Multifactor Quality Method (MQM) to help requirements analysts to empirically collect system constraints in scenarios based on elicited expert preferences. The method combines quantitative statistical analysis to measure system quality with qualitative coding to extract new requirements. The method is bootstrapped with minimal analyst expertise in the domain affected by the quality area, and then guides an analyst toward selecting expert-recommended requirements to monotonically increase system quality. We report the results of applying the method to security. This include 550 requirements elicited from 69 security experts during a bootstrapping stage, and subsequent evaluation of these results in a verification stage with 45 security experts to measure the overall improvement of the new requirements. Security experts in our studies have an average of 10 years of experience. Our results show that using our method, we detect an increase in the security quality ratings collected in the verification stage. Finally, we discuss how our proposed method helps to improve security requirements elicitation, analysis, and measurement.

Index Terms—user study; vignettes; scenarios; security requirements; requirements elicitation, qualitative analysis, context.

I. Introduction

Companies rely on security experts to evaluate system security and determine appropriate mitigations [5, 10, 11]. Despite the abundance of requirements that are available in security checklists and control sets, such as the NIST 800-53 control set [21], security analysts continue to rely on their own experience and background knowledge when analyzing system security [10, 11]. Checklists, which apply to systems, generally, often lack the context needed to assess the threat. For example, Haley et al. assert that it is more feasible to assess security risk and reason about satisfaction of a security requirement in narrow context [8]. Claiming that a negative event is never going to happen is difficult without being explicit about one's trust assumptions [8]. Moreover, mapping the checklist requirement to threat scenarios or to other requirements is usually done by the analyst. In addition, security requirements are not independent, they work together in composition with different priorities and dependencies [5].

Hibshi et al. previously examined the effect of context and requirements composition on security requirements expert ratings [10]. They proposed to use factorial vignettes, wherein requirements and system constraints are variables in a scenario description. That work is limited, since the vignettes were only applied narrowly to website access, there was no guidance on how to select vignette variables , and new requirements were not evaluated for security impact. Also, the ratings were elicited from graduate students, and not security professionals [10].

The goal of this work is to address the above challenges by incorporating the prior technique with the aim of monotonically increasing the system quality. In this paper, we introduce the Multifactor Quality Measurement (MQM) method with an application to security using scenarios across four domains: networking, operating systems, databases and web applications. Further, we show how to empirically improve the scenarios by selecting new requirements that experts suggested would help increase the quality. The MQM method consists of three steps:

- *Bootstrapping:* an analyst selects an initial vignette and a domain as a starting point, and run user experiments with experts to collect quality assessments and elicit additional requirements to improve the quality. The experiments use the factorial vignettes design [28, 33], an approach that is well used in social sciences to elicit human judgments and that had been used in our prior work [10]. This method uses vignettes or scenarios that include discrete factors that can be manipulated to measure differences in judgment as factors change to different levels.
- *Analysis:* next, the analyst consolidates elicited requirements into several improvements that are believed by the experts to increase the overall quality.
- *Verification:* in this optional step, the analyst verifies whether elicited requirements improve overall quality, or whether there are gaps remaining in quality achievement. This step can be run later to assess whether quality has changed.

The paper is organized as follows: in Section II, we review related work, which is the quality studied in this paper; in Section III, we describe the MQM; In Section IV, we present our empirical evaluation method; in Section V we present results; in Section VI, we discuss threats to validity; and in Section VII, we discuss our findings and summarize future work.

II. Related Work

Scenarios are used in requirements elicitation, validation and analysis [14, 32]. Scenario-based techniques have been argued to provide richer details needed in analyzing dependencies between system components and the environment when modeling human uncertainties [32], and in eliciting actual user needs and unforeseen requirements [25, 32].

Scenarios can either originate from the stakeholders' real practices before the system is designed [32], such as by pursuing the inquiry cycle model [25]; or they can originate from the system's specifications and design [32], such as use cases [7, 13], misuse cases [2] and Secure Tropos [18, 20]. Our factorial vignette-based approach uses scenarios to describe an environment that mimics reality to the security analyst to discover dependencies among requirements and elicit previously unforeseen requirements that mitigate threats. Unlike the inquiry cycle model that searches the requirements space in multiple directions by asking *what*, *how*, *where*, *when* and *why* questions [25], our approach first collect answers to some of these questions and then shows these answers to expert. Then, the quality is measured to evaluate the strength of the answer toward affecting the overall system behavior, and this is obtained by experts' evaluations.

Van Lamsweerde and Willemet [15] define scenarios as a temporal sequence of interaction events between agents of a system composed of software agents, human agents, and hardware devices [15, 16]. We found this definition common among requirements researchers who use scenarios that originate from a system's specification [16, 18, 20, 14]. Van Lamsweerde and Willemet use operational scenarios to infer goals and generate specifications in the KAOS goal-based specification language [15]. In their approach, they express scenarios using event trace diagrams, and they propose using a formal method based on inductive-learning to refine and analyze the goal specification by covering positive scenarios and eliminating negative scenarios with undesired behavior [15]. In addition, stakeholders are not directly involved with the evaluation of scenarios and their specifications [15]. Letier et al. later found that classifying scenarios as positive or negative is difficult due to the lack of domain knowledge e and conflicting stakeholder viewpoints [16].

Scenario-based analysis has been used in the Secure-Tropos framework to analyze security requirements in multi-agent systems [18, 20]. Security attack scenarios are created by requirements analysts, and then inspected to find violations of the syntax or inconsistences between the scenarios and the models. Finally, the requirements analyst runs test cases generated from the security attack scenarios to test a system's vulnerability during an attack [18, 20]. A framework like Secure-Tropos could be extended in future research using our proposed MQM method to evaluate attack scenarios.

Scenario-based research in requirements engineering, including the work noted above, share a common feature: the ad-hoc starting point of scenario creation, which is generated by the analyst, is then re-inspected or refined. In MQM, a similar approach is adopted, which we call bootstrapping, wherein an analyst designs an initial scenario from their limited domain knowledge. Because the method guides the analyst toward increases in quality, this approach is preferable to an otherwise unbounded process with no clear guidance on where to stop generating more scenarios. Letier et al. proposed a scenario-based technique for requirements analysis and they indicate that adding both positive and negative scenarios results in large unstructured models [16].

Potts distinguishes *abstractionism*, where researchers rely on formal models; and *contextualism*, where the context of the system is well understood before deriving requirements [23]. Potts argues that abstractionist approaches use simplified models of a phenomena that leave out stakeholders' needs and requirements. Contextualist approaches use rich details resulting in creating systems that reflect the context of use and satisfy the stakeholder needs in the short term, but can be expensive and time consuming to progress or scale the design over time. Potts suggests that to build more useful systems, researchers should focus on approaches that integrate the two philosophies. The integration should adopt a strong committed view [23, 24] in which, intangible phenomena are not explicit [17, 23], but rather they are implicit in the stakeholders' interpretation of the domain [23, 24]. For example, business processes are not tangible, as they exist in the stakeholders' interpretation of a business [23]. In our work, we acknowledge that security requirements in practice exist in the security analysts' interpretation of a system and their judgment of the situation, and that organizations rely on security experts' recommendations [11]. The MQM method introduces structured scenario design using textual templates, which is different from strict contextual designs such as natural inquiries [17, 24] that rely on rich descriptions, but could be time consuming.

III. Multifactor Quality Measurement

We now describe the Multifactor Quality Measurement (MQM) method for eliciting system constraints that affect overall quality. In prior work [10], we presented an empirical evaluation of using factorial vignettes for collecting security and found it to be effective. In this paper, we are integrating the technique into a framework, the MQM, that can be extended and reused outside of security. Figure 1 shows the different stages of the MQM. In addition, we address the limitations of prior work in the following way:

1. We evaluate the MQM across four domains: networking, operating systems, databases and web applications. In prior work [10], only one domain was evaluated (computer user surfing the web).
2. Participants are put in an expert role in the scenario (e.g. network administrator)
3. We recruit security experts from industry and government.

We now describe each phase of the MQM.

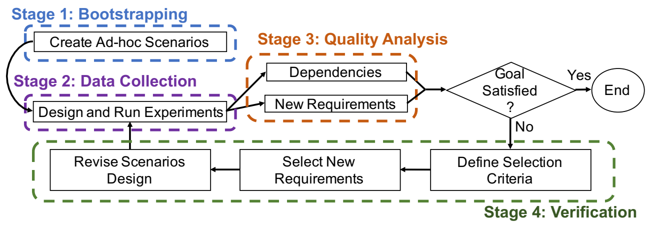

Fig. 1. The Multifactor Quality Measurement (MQM) Method

A. Stage 1: Bootstrapping

During bootstrapping, an analyst first chooses the quality to evaluate, and then the analyst chooses an initial scenario that describes a cohesive system viewpoint [22]. The ad hoc scenario is selected by the analyst who might have limited knowledge, because the MQM will collect empirically measured improvements in this stage. This scenario is a text-based system description that includes the ways people interact with the system. We show an example scenario template in Figure 2.

Figure 2 shows a template from the web applications security domain that consists of variables preceded by the ($) sign. A variable in the scenario is a security requirement category. The variables are replaced by different values that correspond to constraints on the system. The manipulation of variables and their values allows the analyst to generate different instantiations of the template, called vignettes, which will increase the number of scenarios that can be evaluated at one time. The `$WebAuth` variable represents the type of authentication used in the web application and it can take one of many values. To illustrate, we consider two extremely different values: "basic authentication," which is a weak form of web-based authentication, or "form-based authentication using encrypted credentials stored in a database," which is stronger. Similarly, the `$StoredUserData` variable represents how the user input is being collected, and could take the values: "collect user-supplied content from GET request," or "require CSRF tokens and escape and validate user-supplied content from POST requests before storing;" and again, the latter value is stronger than the former.

You are a website administrator responsible for securing a web app against cyberattacks. Currently, you are evaluating the following settings:

- The web app performs **$WebAuth**.
- The web app will **$StoredUserData** in a database for display to other users.

The Cross-Site Request Forgery attack is a serious security concern. Please answer the following questions with regards to mitigating this threat.

Fig. 2. Example Scenario Template from the Web Applications Domain

Study participants are asked to rate the adequacy of the overall security of the scenario on a 5-point scale where point 1 is labeled "inadequate", point 3 is labeled "adequate" and point 5 is labeled "excessive." This generates the `$Overall` dependent variable. Similarly, we ask users to provide ratings for the individual security requirements in the scenario, which generates a dependent variable for each rated requirement. For example, the web applications study has the `$WebAuthRating`, and the `$StoredUserDataRating`, which are the dependent variables representing experts' ratings of the `$WebAuth`, and `$StoredUserData`, respectively.

After creating the initial ad-hoc scenario, the analyst decides the number of factors and factor levels in the scenario:

- Factors per domain*:* a domain could have its own subset of factors, with the possibility of having factors that are shared among different domains. The factors often correspond to categories of system constraint e.g., passwords, authentication type, etc.. In addition, factors may, but do not necessarily have to, cross multiple domains, e.g., passwords affect databases, networks, and systems.
- Levels per factor: how many levels will be manipulated. The levels, which correspond to technically specific interpretations of the factor, can be chosen as high or low levels. The goal is to choose levels that experts can distinguish to measure an effect or interaction among different levels. For example, if password complexity has high and low levels, we can measure whether password complexity affects overall security adequacy in conjunction with other security constraints.

Deciding on the number of factors depends on the quality of interest, the cost of running the surveys, and the estimated number of experts available to rate the scenarios against the quality of interest. An analyst would need to conduct a priori statistical power analysis to decide on the right number of factor/level combinations. Initial pilot studies and focus groups can also help with the design decisions in the bootstrapping phase as it would help eliminate unrealistic factor and level combinations [10].

We are not limited to one template, in addition to the web application template shown in Fig. 2, we describe in Section IV how to generate more templates and integrate factors and levels for three more security domains.

Domain experts may suggest additional unforeseen requirements that would improve the measurements. An analyst could elicit new expert requirements from experts to improve the measurements. For example, security experts could provide more mitigation that would increase the adequacy ratings, so, we ask experts to list additional mitigations that they believe will increase security.

B. *Stage 2: Data Collection*

Once the scenarios are ready, the analyst finalizes the design of the overall experiment. This includes deciding which factors are between-subject or within-subject factors. The analyst in this stage decides on how to operationalize the survey: recruitment methods (e.g. in person, online, mailing lists), tools to be used, and whether expertise screening questions are needed (e.g. knowledge tests, demographics). Finally, the analyst deploys the survey and starts data collection.

C. *Stage 3: Quality Analysis*

In this stage, the analyst uses regression analysis to discover the weights of the factor levels (e.g., `$WebAuth` and the `$StoredUserData`) and to discover any interactions among the variables. The priorities of requirements are decided based on the weight of the coefficient. The type of regression (e.g. linear, multi-level) depends on the study design (within-subject vs. between-subject effect). Linear regression is used when there is no within-subjects effect in the data, while multi-level modeling is used if there is at least one within-subject factor. Next, the analyst classifies the experts new requirements into broader categories and links these to the factors/levels in the scenario.

The collected new expert requirements mitigations are expressed in natural language. The problem with natural language statements is that different experts could describe the same requirement using different words and phrases. As a first step, requirements are coded using short phrases (concept labels), an open coding grounded analysis approach [10, 29]. Then, the analyst categorizes the requirements using a more abstract security concept. For example, mitigations coded as *password salt* and *stronger password*, are grouped under *passwords*; and *input sanitization* and *input validation* are categorized under *SQL injection mitigations*.

After first-cycle coding and categorization, a second-cycle coding is conducted [29], where requirements are linked to the factor levels that they appear in, which would help to filter the requirements that we anticipated to appear vs. new unanticipated requirements. For example, in the network study, there are scenarios with insecure *Dematerialized Zone* (DMZ) configuration and a more secure split-DMZ configuration. Mitigations that suggest better network segmentation are linked to the level of the DMZ level shown in scenarios where the mitigation was elicited. If associated with the weaker DMZ, then this makes the mitigation anticipated, but if associated with the stronger DMZ, then that means there are further segmentation configurations for the network and DMZ that was not anticipated in the scenario.

In addition, each requirement is assigned one of the following codes: refinement, if the requirement refines the dimension by extending its functionality; a reinforcement, if the requirement adds auxiliary quality not directly related to the dimension; and a replacement, if the requirement replaces the dimension.

Upon completion of analysis, the analyst decides to either stop and be satisfied with the data collected, or continue to the next stage: verification. *Verification* is an expensive step that the analyst could pursue if the results show rich data that needs further verification, and stop once they reach saturation. By saturation, we mean no new requirements are being collected and the analyst continues to see the same statistical results (e.g. same effect, same dependencies among the variables).

D. *Stage 4: Verification*

Based on the output of stage three, the analyst defines a selection criteria and heuristics that will guide the requirements selection process. For example, to ensure monotonically increasing quality, an analyst may only select requirements that would increase the quality of interest in the next scenarios.

In our series of security experiments, our goal is to increase security adequacy. Hence we define the following criteria:

- For each domain, select two categories from second cycle coding with the highest number of requirements within the category.
- For each category, select the requirements with highest frequency that appear even in vignettes where the level of the requirement is strong.

In the verification stage, the requirements evaluated in the bootstrapping stage are assigned a fixed level, which is the strong security level. By fixing these levels, the effect of unanticipated requirements becomes the focus of measurement.

Then, the analyst will repeat steps from stages two and three to verify whether the new set of requirements affects the quality measurements as intended. To exit the iterative process of the MQM, the analyst establishes an end goal to be achieved.

IV. Experimental Evaluation of the Method

In this section, we explain the research approach that we use to evaluate the MQM on security-specific domains.

A. *Stage I: Bootstrapping*

For this stage, we select the initial security vignette that is needed to design and run a user study to collect from security ratings of security requirements from experts.

We selected four different security domains and we ran four user studies one for each domain. Figure 3 shows the text template used for all four studies. The values for the variables shown Fig. 3 are changed depending on the user study domain. Table I lists the variables used for security requirements in the four domains and their levels. For example, the `$Domain` is replaced with either network, systems, database, or web applications. The factors (`$Factor1, $Factor2`) are replaced with different sets of security requirements factors for each domain (*factor* column in Table 1). Within a domain, the factors are manipulated with different values (levels) to generate the values for the user study corresponding to that domain.

Fig. 3. Text template for the four security domains

```
 A popular online retailer offers a wide variety
of products for purchase.  User information in the
company's databases includes consumers' credit card
information for purchasing products in the future.
  You are a $Domain administrator for the retailer
  who is responsible for securing the $Domain
  against cyberattacks. Currently, you are
  evaluating the following settings:
  - $Factor1
  - $Factor2 …
  The $Threat attack is a serious security concern.
  Please answer the following questions with regards
  to mitigating this threat.
```

B. *Stage II: Data Collection*

Each vignette has a different combination of variable levels which generated 12 unique vignettes (study conditions) for each of the network (2x3x2), systems(2x2x3), and databases studies (2 x3x2), and 8 vignettes for the web applications study(2x4). We decide to choose one factor in each study to have a within-subjects effect. This approach increases power at smaller sample size (security experts are scarce [12]). A participant will evaluate 4 vignettes: two domains with two vignettes in each domain. Within a domain, the two vignettes will vary by the within-subjects 2-level factor. The variables shown in bold in Table I are within-subject variables: each participant has seen all the levels of that variable; the remaining variables are between-subject variables where each participant was exposed to one level only of that variable. This results in a mixed-effect design.

Upon completion of the security ratings, participants are asked to take a security knowledge test (14 questions); and answer demographics questions (e.g. gender, age, experience, etc. see Section V). It is recommended to place background and demographics questions at the end of surveys to avoid potential bias and to increase participants' response rate [27].

We targeted security experts who attended the SANSFIRE 2016 conference at Washington, DC. The SANS is a security research and education company that offers security training and certification to government and industry security analysts [31]. We compensated each participant with a $25 Amazon gift card.

C. *Stage III: Analysis*

We will explain below the method of analysis for the quantitative and the qualitative data collected in stage 2.

1) *Analysis for Dependencies*

We analyze the quantitative data using multilevel modeling. Multilevel regression models can better handle the mixed effect in the study design (between-subject and within-subject effects) [6]. Each dependent variable generated from user ratings (see Section IV.A) is analyzed using multi-level regression.

For the security knowledge test, we use a `$Score` variable, which is an independent exploratory variable assigned an integer value equal to the percentage of correctly answered security questions.

We use: R [26] with the lme4 [1] and SJPlot [19] statistical packages, and the G*Power tool [3] for the power analysis.

TABLE I. USER STUDY SECURITY DOMAINS AND THEIR CORRESPONDING REQUIREMENT VARIABLES

Domain	Threat	Factor	Level Code	Level description
Network	Man-in-the-middle	**$NetworkAccess**	onsite	Onsite access using Ethernet
			offsite	Offsite external access through a secure VPN
		$NetworkAuth	simp6	A standard 6-digit password
			comp16	16-char password that must include an uppercase letter, lowercase letter, a symbol, and a number
			multi8	An 8-character alphanumerical password and a one-time password sent to a mobile phone
		$DMZ	allnosplit	DMZ contains the webserver, app server and the database server.
			split	DMZ contains the front-end webserver and the app server. The DB server is behind the firewall on the internal network. The app server communicates with the DB over a VPN.
Systems	malware	**$SocialMedia**	permit	Workstations permit access to social media sites
			prohibit	Workstations prohibit access to social media sites
		$AdminPriviledges	noauth	Prior to installing new software, employees who are local system administrators, are not required to re-authenticate
			auth	Prior to installing new software, employees are required to re-authenticate
		$VirusScanner	files	Workstations has programs to scan files against known malware signatures
			filesmem	Workstations has programs to scan memory and files against known malware signatures
			filesmempro	Workstations has programs to scan memory, files and processes against known malware signatures
Database	Privilage escalation	**$DBAccess**	extserver	User accounts and access control are handled by SQL table authentication
			sqlauth	User accounts and access control are handled by Windows Active Directory
		$DBMonitor	available	Database activities are logged
			needed	Database activities are logged, and inspected as needed (e.g., to examine a certain incident)
			month	Database activities are logged, and inspected each month by a trained auditor
		$Error	User	Errors are handled by notifying users who can then report the error message, as needed
			nouser	Errors are handled by logging the error message with no external notification to users
Web applications	Cross-site-request forgery	**$WebAuth**	basic	Basic authentication
			form	Form-based authentication using encrypted credentials stored in a database
		$StoredUserData	get	store user-supplied content from GET requests
			post	store user-supplied content from POST requests
			cpost	require CSRF tokens for user-supplied content from POST requests before storing
			cescpost	require CSRF tokens, escape and validate user-supplied content from POST requests before storing

TABLE II. USER STUDY SECURITY DOMAINS AND THEIR CORRESPONDING ADDED REQUIREMENT VARIABLES

Domain	Threat	Factor	Level Code	Level description
Network	Man-in-the-middle	$MFA	enabled	There is a one-time password sent to a mobile phone
			disabled	There is no further tokens or one-time passwords sent to mobile-phones
		$DBSegment	empseg	The DB Server is placed on a special admin segment separate from the employee network
			sepseg	The DB Server is placed on the same segment with the employee network
System	malware	$SWInstallation	notest	Admins are specific IT professionals who can install any new SW with no further testing
			test	New software must be tested and approved prior to installation
		$MalwareTools	enabled	Heuristic-based and behavioral-based malware-detection tools are enabled
			disabled	Heuristic-based and behavioral-based malware-detection tools are disabled
Database	Privilage escalation	$SIEM	siem	A trained IT auditor inspects logs with a specialized SIEM (Security information and event management) tool that the company installed for log analysis and management.
			nosiem	A trained IT auditor inspects logs without the assistant of costly SIEM tool
		$Notification	enabled	Admins are automatically notified when errors occur
			disabled	No notification sent to admins
Web Apps	Cross-site-request forgery	$InputValidation	client	on the client-side
			server	on the client-side, followed by input sanitization on the server-side
		$SOP	verify	In addition to the CSRF token, HTTP standard headers are examined for same origin
			noverify	The CSRF tokens are robust. No need to verify Same Origin on the server side

D. *Stage IV: Verification*

Based on the selection criteria defined in Section III, we select two new requirements from the reinforcement category for each security domain. The new generated scenarios will keep the bootstrapping requirements, and include new variables for the new reinforcement requirements. Since the goal is to increase security ratings, we fix the levels for the bootstrapping requirements at the strongest level. For the new requirements, we use a weak and a stronger level to test their effect in improving security ratings. Hence, each new study domain had a 2x2 factorial design (2 new variables with 2 new levels each). Table II lists all the added requirements and their levels. After deciding on the new requirements and the redesign on the new vignettes, we ran the user experiments using the same protocol from the bootstrapping stage, but with the following changes:

- Recruitment: we re-invited security analysts that we previously recruited for the bootstrapping stage and for other security-related studies by using the emails they provided *to opt-in for future studies.* We sent each participant a unique one-time code to be used to access the online survey.
- *Experiment set-up:* we set up the user experiment such that each participant sees one vignette from each domain, so the experiment has a between-subject design (no-mixed effects).
- *Statistical analysis:* since the new design is between-subject with no mixed-effect, we use linear regression for analysis

V. Results

We report our sample demographics, and statistical results from the bootstrapping and verification stages.

A. *Descriptive Statistics from the Bootstrapping Stage*

The bootstrapping stage aims to collect ratings and new requirements for an ad hoc vignette. In this stage, we recruited 69 security participants. Table III summarizes our sample demographics, and the participants' performance on the security knowledge test. Participants have an average of 10 years of experience. The number of responses for each domain is: 39, 30, 49, and 21 for networking, operating systems, databases, and web applications, respectively (each participant was randomly assigned to two vignettes from two domains, see Section IV).

B. *Dependency Analysis from the Bootstrapping Stage*

The `$OverallRating` represents the experts' security rating of the scenario based on the composition of the requirements. We show an example of the regression equation for the web applications domain. Equation 1 is our additive regression model with a random intercept (ϵ) grouped by participant ID.

$$\$OverallRating_{webapp} = \alpha + \beta_w \$WebAuth + \beta_s \$StoredUserData + \epsilon \quad (1)$$

The additive model is a formula that defines the `$OverallRating` in terms of the intercept (α) and a series of components. Each component is multiplied by a coefficient (β) that represents the weight of that variable in the formula. The formula in Eq. 1 is simplified as it excludes the dummy (0/1) variable coding for the reader's convenience. We use the same formula for each domain, but we replace the independent variables corresponding to the factors in that domain. We follow a similar model for the individual requirements ratings. For example, Equation 2 below is the additive regression model for `$WebAuthRatings` variable.

$$\$WebAuthRating_{webapp} = \alpha + \beta_w \$WebAuth + \beta_s \$StoredUserData + \epsilon \quad (2)$$

TABLE III. Bootstrapping Study: Demographics

Description		Participants *Number*	*Percentage*
Gender*	Male	59	86%
	Female	7	10%
Years of Experience* (Mean=10)	Less than 2	9	13%
	2 – 5 years	15	22 %
	6 – 10 years	15	22 %
	11 – 15 years	9	13%
	16 – 20 years	13	19%
	more than 20 years	5	7%
Job Sector*	Industry: non-research	24	35%
	Government: non-research	22	32%
	Industry: research	5	7%
	Academia	5	7%
	other	9	13%
Took academic classes in security		39	57%
Took job training in security		54	78%
Self-taught security knowledge		54	78%
Job roles	Security analyst	46	67%
	Other – IT security related	6	9%
	Other – IT related	13	19%
	Other – Non IT	4	6%
Highest Degree Completed	Bachelor's degree	31	45%
	Masters graduate degree	17	25%
	High school or equivalent	8	12%
	Some college, no degree	7	10%
	Associate degree	5	7%
	PhD degree	1	<1%
Security Knowledge Score	Scored above 60%	18	26%
	Scored between 40% and 60%	40	58%
	Scored below 40%	11	16%

*A few participants did not answer this question

We report the significant results of our bootstrapping stage data in Table IV[1]. We use the variable and level codes shown in Table I. For each security domain, we establish a baseline level for factors in that domain. The intercept (α) is the value of the dependent variable when the independent variables are at their baseline values. The baseline levels for each domain are shown in Table IV. Table IV also shows the *coefficient estimates* (*Coeff. Est.),* which show by how much the security requirement level increased or decreased the mean rating of adequacy.

For the networking domain study, we found a significant contribution of the three network factors (`$NetworkAccess`, `$NetworkAuth`, and `$DMZ`) for predicting the `$OverllRatingNetwork` ($\chi2$ (7) =11.3, p=0.022), over the null model (without the factors). Table IV shows a significant effect from multifactor authentication for the network authentication requirement (coded `multi8`, see Table I), increasing the ratings over the intercept (1.83) by approximately one point (0.96*)* on the adequacy scale (almost adequate). Among all networking

[1] Full dataset and regression results are available online: http://gaius.isri.cmu.edu/dataset/mqm17/

scenario requirements, only `$NetworkAuthRating` shows a significant effect (χ2 (4) =18.3, p=0.001) (see Table IV).

TABLE IV. SIGNIFICANT MULTILEVEL REGRESSION RESULTS FOR THE BOOSTRAPPING DATA

Dependent Variable (DV)	Independent Variable (IV) - level	Coeff. Est.	Std. Error
Networking	IVs: $NetworkAccess+$NetworkAuth+$DMZ		
baseline	*offsite+ comp16 + allnosplit*		
OverallRating	*Intercept (baseline)*	*1.83****	*0.28*
	NetworkAuth - (multi8)	0.96**	0.34
Network Auth-Rating	Intercept (baseline)	2.28***	0.30
	NetworkAuth - (multi8)	0.75*	0.36
	NetworkAuth - (stand6)	-0.72*	0.36
Operating Systems	IVs: $SocialMedia+$AdminPriviliges+$VirusScan		
baseline	*permit+ auth + files*		
OverallRating	*Intercept (baseline)*	*2.2****	*0.39*
	AdminPrivileges- noauth	-0.95*	0.37
SocialMedia Rating	*Intercept (baseline)*	*2.06****	*0.40*
	SocialMedia- prohibit	1.13***	0.19
AdminPriviliges -Rating	*Intercept (baseline)*	*2.31****	0.43
	AdminPrivileges- noauth	-1.33***	0.41
VirusScan-Rating	*Intercept (baseline)*	*2.61****	0.35
	VirusScan - filesmemoryprocesses	0.89*	0.37
Database	IVs: $DBAccess+$DBMonitor+$Error		
baseline	*extserver + available + nouser*		
OverallRating	*Intercept (baseline)*	2.89*	0.33
interaction terms	Error - user	-1.35**	0.45
	DBAccess - sqlauth * DBMonitor - month	-0.60**	0.29
	DBAccess - sqlauth * DBMonitor - needed	-0.57*	0.28
	DBMonitor - month * Error - user	1.33**	0.60
ErrorRating	*Intercept (baseline)*	2.8***	0.28
	Erroruser	-0.98***	0.27
Web Applications	IVs: $WebAuth+$StoredUserData		
baseline	*basic + cescpost*		
OverallRating	*Intercept (baseline)*	2.36***	0.21
	StoredUserData - get	-0.73***	0.25
	StoredUserData - post	-1.32***	0.29
	StoredUserData - cpost	-0.70***	0.29
WebAuthRating	*Intercept (baseline)*	2.04***	0.26
	WebAuthform	0.76***	0.21

(*p≤.05 **p≤.01 ***p≤.001) In the database domain, we see an effect for the interaction terms of the regression model for the overall security rating (χ2 (9) =20.7, p=0.01). Reporting errors to users (`Error - user`) decreased the security rating by more than a point, but when the reporting errors to users are combined with a more frequent logging mechanism (`DBMonitor - month`) the rating increases over the baseline.

C. *New Requirements from the Bootstrapping Stage*

After text cleanup and preparation, participants provided a total 550 mitigations that we classified into 55 categories and 187 sub-categories. Table V shows the top five categories for each domain based on number of occurrences (*Freq.*). The table shows how some categories appear in multiple domains (e.g. accounts/access control), while other categories were unique to a security domain (e.g. SQL injection mitigations).

D. *Descriptive Statistics from the Verification Stage*

The verification stage aims to evaluate to what extent the new requirements increase security. We sent 100 email invitations, and received 45 expert responses (45% response rate). Survey Gizmo, a large online surveying platform reports that internal employee surveys receive a 30-40% response rate on average and external surveys receive an average of 10-15% [4]. Compared to the bootstrapping stage, respondents to the verification stage scored higher on the security knowledge test ($\text{Mean}_{\text{Bootsrapping}}$ = 52%, $\text{Mean}_{\text{Verification}}$ = 60%).

TABLE V. TOP FIVE MITIGATIONS CATEGORIES

Networking		**Operating Systems**	
Category	*Freq.*	*Category*	*Freq.*
Passwords	29	Accounts/Access Control	59
Segmentation	20	Software Installation	21
Authentication	17	Social Media	17
Firewalls	6	Malware Detection	13
Certificates	6	White/Blacklisting	12
Databases		**Web Applications**	
Category	*Freq.*	*Category*	*Freq.*
Logs	74	Authentication	14
Accounts/Access Control	68	SQL Injection Mitigations	9
Error Handling	31	Web App Protections	9
Monitoring	10	Accounts/Access Control	4
Authentication	8	Testing	4

TABLE VI. VERIFICATION STUDY: DEMOGRAPHICS

Description		**Participants** *Number*	*Percentage*
Gender*	Male	43	96%
	Female	1	2%
Years of Experience* (Mean=9)	Less than 2	1	2%
	2 – 5 years	14	31 %
	6 – 10 years	16	36 %
	11 – 15 years	8	18%
	16 – 20 years	4	9%
	more than 20 years	2	4%
Job Sector*	Industry: non-research	14	31%
	Government: non-research	12	27%
	Industry: research	2	4%
	Government: research	6	13%
	Academia	3	7%
	other	7	16%
Took academic classes in security		34	76%
Took job training in security		40	89%
Self-taught security knowledge		37	82%
Job roles	Security analyst	30	67%
	Other – IT security related	4	9%
	Other – IT related	4	9%
	Other – Non IT	4	9%
Highest Degree Completed	Bachelor's degree	12	27%
	Masters graduate degree	24	53%
	High school or equivalent	2	4%
	Some college, no degree	4	9%
	Associate degree	1	2%
	PhD degree	1	2%
Security Knowledge Score	Scored above 60%	20	44%
	Scored between 40% and 60%	21	47%
	Scored below 40%	4	9%

*A few participants did not answer this question

E. *Statistical Analysis from the Verification Stage*

We now review the linear regression results from the verification stage, before comparing the security ratings obtained from bootstrapping and verification.

1) *Regression Analysis of Verification Study Data*

Recall from Section III, the MQM uses linear regression to analyze the results of the vignette surveys responses. The independent variables in the regression formula are the requirements variables shown in Table II to verify the effect of the new requirements on the security ratings. We now report the regression results for each security domain.

a) Networking: the regression model shows that different levels of the new requirements variables `$MFA`, and `$DBSegment` do not significantly predict the `$Overall` security rating, because the regression model of `$Overall` ratings as a function of the `$MFA`, and `$DBSegment` did not show any significance over the intercept-only model (F(2,39) = 1.595, p=0.2). Hence, the `$Overall` mean, which is the intercept-only model is a better predictor of the overall security ratings for the networking study. The result is similar for the regression models constructed for the `$NetworkAccessRating`, `$NetworkAuthRating`, `$DMZRating` with: (F(2,42)=1.2, p=0.3), (F(2,42)=0.04, p=0.9) and (F(2,42)=0.5, p=0.6), respectively. The `$MFA` variable that represent multifactor authentication is shown to be a good predictor of the experts `$MFARating` (F(2,42)=5.3, p<0.01). Scenarios that include multifactor authentication show an increase of 0.85±0.27 (standard error) on the `$MFARating` scale (p<0.001). Similarly, scenarios where the database is in a separate segment (`$DBSegment`) shows a significant increase (p<0.001) in the `$DBSegemtnRating` by 1.4±0.30 (F(2,41)=11.4, p<0.001). *Operating System:* The regression for `$Overall` ratings as a function of `$SWInstallation`, and `$MalwareTools` show significance (F(2,41)=4.57, p=0.02) over the intercept-only model. When inspecting the coefficients, only the intercept and `$MalwareTools` show significant effects. Enabling heuristic-based and behavioral-based malware-detection tools show a significant increase (p=0.02) in the `$Overall` ratings by 0.53±0.22 and also show a significant increase (p<0.001) in the `$MalwareRating` by 1.68±0.16; thus, `$MalwareTools` is a good predictor of the `$MalwareRating` (F(2,42)=61.26, p<0.001). `$SWInstallation` is found to be good predictor of the `$SWInstallationRating` (F(2,42)=35.25, p<0.001). Scenarios that include testing new software prior to installation (`$SWInstallation`) show a significant increase of 1.5±0.18 of the `$SWInstallationRating`. We found no significant effect for the regression models constructed for the `$SocialMediaRating`, `$AdminPriviligesRating`, `$VirusScanRating` with: (F(2,42)=1.33, p=0.3), (F(2,42)=1.63, p=0.2) and (F(2,42)=1.45, p=0.2), respectively. We also found no significant effect for the interaction terms.

b) Databases: The regression model of `$Overall` ratings as a function of `$SIEM`, and `$Notification` show no significance (F(2,38)=1.06, p=0.35) over the intercept-only model. Except for `$DBMonitorRating` and `$NotificationRating`, no significant effects are found for the requirements ratings in the database scenarios. Database scenarios that include using a specialized SIEM (security information and event management) tool, show a significant (p=0.009) increase of 0.54 ±0.20 on the `$DBMonitorRating`. The `$SIEM` shows significance in predicting the `$DBMonitorRating` (F(2,42)=3.8, p=0.03). Similarly, `$Notification` is a good predictor of the `$NotificationRating` (F(2,42)=24.29, p<0.001). Scenarios that include notifying admins about errors show a significant (p<0.001) increase of 1.48±0.22 on the `$DBMonitorRating`.

c) Web Applications: Except for the regression model constructed for `$SOPRating`, which rates the same origin policy, no significant effects are found for the `$Overall` rating nor for all other requirements in this scenario. For the `$SOPRating`, it was not the `$SOP` variable that significantly affected this rating, but the `$InputValidation`. Scenarios that include validating the client's input on the server-side, show a significant (p=0.007) increase of 0.9±0.32 on the `$SOPRating`. The `$InputValidation` show significance in predicting the `$SOPRating` (F(2,39)=4.03, p=0.03).

The major takeaway is that the *intercept-only* model is sufficient to explain the outcome dependent variable. The significance of the intercept-only model means that we can rely on using the *means* of the dependent variables to explain the observations in the data. For the security analyst, this means that varying levels of new factors did not show significance, but we cannot remove the factors from the model. We will explain this further as we show the mean values in the next section.

2) *Comparing the Security Ratings*

In Table VII, the mean ratings in the verification stage are higher than the bootstrapping stage, except for the overall rating for database, which has a slightly lower average than the bootstrapping stage. Table VII also shows that some variables increased more than others, for example, the `$OverallRating` for *Networking* only increased by 0.20, while the `$NetworkAuthRating` increased 4.5 times by 0.90. Despite the ratings increase, the values in Table VII also indicates that all averages are close to adequate (3.0 on the 5-point scale). The standard deviation of all the ratings ≤1.

TABLE VII. COMPARISON OF EXPERTS' SECURITY RATINGS

Rating Variable Name	**Bootstrapping Stage**	**Verification Stage**
	Mean Rating	***Mean Rating***
Networking		
OverallRating	2.37	2.57
NetworkAccessRating	2.70	3.09
NetworkAuthRating	2.32	3.22
DMZRating	2.53	2.82
Operating Systems		
OverallRating	2.10	2.70
SocialMediaRating	2.60	3.13
AdminPriviligesRating	1.74	3.07
VirusScanRating	2.73	2.80
Databases		
OverallRating	2.51	2.34
DBAccessRating	2.62	2.71
DBMonitorRating	2.56	3.00
ErrorRating	2.25	2.60
Web applications		
OverallRating	1.80	2.62
WebAuthRating	2.05	2.69
StoredUserDataRating	1.86	3.07

VI. THREATS TO VALIDITY

External validity concerns how well results generalize to the population [30]. Our target population is security experts and we recruit security professionals who attend security conferences.

To assess security expertise, we measured years of experience (mean=10.0 years) and we conducted a security knowledge test that included technical questions about how to configure file permissions, network firewalls, etc.

Internal validity is the degree to which a causal relationship can be inferred between the independent and dependent variables [30]. We randomize the assignment of participants to conditions, and we randomize the presentation order of scenarios. Based on our pilot results, we limited the number of vignettes shown to four vignettes per participant to reduce fatigue. We ran the verification study seven months after the bootstrapping stage to reduce learning effects.

Construct validity is the degree to which a measurement corresponds to the construct of interest [30]. In each scenario, we present one-sentence definitions for the security level terms inadequate, adequate, and excessive, to encourage participants to interpret the label levels, similarly. The label name choice was evaluated in separate prior studies by Hibshi and Breaux [9, 12].

Increasing power in user experiments reduces Type II errors (false negatives). We increase our power in the bootstrapping stage by using repeated measures within-subject effect, and analyzing the data with multi-level modeling, which assigns a random intercept for each subject and hence, limits the biased covariance estimates [6]. For a power of 80% or above, we estimate a sample size of 30 participants for the networking, operating systems, and database scenarios and 24 participants for the web applications scenario. We achieved higher sample sizes than these minimum estimates. For the verification phase, we estimate 30 participants per domain to achieve at least 80% power, and our actual sample size is 45 participants per domain.

VII. Discussion and Future Work

In this paper, we introduce the MQM method that provides means to empirically elicit and score security requirements from security experts. We now discuss our findings and their impact on the field of security requirements engineering.

Our results show that the mean overall security ratings increased in the verification stage over the bootstrapping stage. This means that experts view the refined scenarios in the verification stage to have higher security adequacy than the original scenarios used in the bootstrapping stage. The results in Table VII also indicate that the average ratings are approximately 3 ±1 (STD) (adequate=3, see Section III). One possible explanation could be that security experts are more conservative when rating security and cannot envision excessive security. Hibshi et al. found that security experts do prefer more conservative security ratings [12].

The regression analysis of the verification stage also shows that the new requirements matters to the analysis, but the individual levels do not vary significantly. While in the verification stage experts report a ratings increase over the bootstrapping stage, the increase cannot be attributed to the new requirements levels. This finding yields two key insights: *security saturation*, wherein it is sufficient to accept new, elicited requirements and a verification stage may not be necessary; and *label bias,* in which the excessive label is unreachable and thus reduces the ability to measure significant differences. We now further discuss these two insights.

Reaching saturation is an important point in empirical research, where analysts receive little new information and thus they can stop iterating through a process. Saturation is also important in practice, because security analysts would prefer a *wish-list* of all possible security mitigations, but it is the financial cost that forces analysts to revise and only choose *what is necessary*. Our results from the verification stage indicate that increasing the requirements from the bootstrapping stage to a stronger level is what is necessary to reach security adequacy. When we increased the requirements to a stronger level, the overall security increased to a point that the two new added requirements with their levels did not necessarily standout in the regression model. This is an effect caused by the combination of strong security requirements in the scenarios tested.

In the bootstrapping stage, we paid $1,725 in gift cards ($6.25 per scenario) to collect evaluations of 44 scenarios from 69 experts, in addition to a $600 overhead which is the cost to send the researcher to Washington D.C. We chose the gift card value based on a $50-hourly rate, which is the average rate for experts shown in our expert-salary data. We find this cost-effective, since we can collect data in one day by flying to one conference venue and with little effort to convince experts to participate. The data analysis took one month, and the surveys for the verification stage were completed in two weeks ($1,425 in gift cards). In contrast, our prior research [11], consisted of 11 expert interviews over a 6-month period, wherein the analysis of the interview transcripts required another 6 months. For an organization to hire security experts to evaluate scenarios or to perform risk-based security analysis, the cost will be more than paying the average hourly rate, due to the added overhead of experts' recruitment and accommodations.

For future research, we may consider combining new requirements and bootstrapping requirements to examine dependencies and investigate combinations with *replacement.* For example, a scenario that replaces a bootstrapping requirement passwords with one-time passwords. In addition to replacements, contrasting weak and strong requirements could add more insights about requirements *tradeoffs*. Recall from Section V.B, reporting errors to users (Error – user) decreased the overall ratings of the database scenario, but the overall ratings improve when the database scenario changes to include a stronger level of monitoring and log inspection. We did not examine such combinations, which may highlight priorities and weights among requirements. For practitioners, such combinations could lead to important questions: e.g., are our data assets worth investing in a costly logging analysis tool, or is it adequate to fix the error reporting mechanism to users?

The security ratings that do not rise above adequate raise a question about the adequacy scale. The adequacy scale was evaluated in a prior study to select the appropriate language labels that explain adequacy [9, 12]. The evaluation examined synonyms for inadequate, adequate and excessive in four scenarios where adequacy perception is skewed by the object being evaluated [9, 12]. Haley et al. proposed a framework to "*determine adequate security requirements for a system*" [8]. What has not been discovered, yet, is whether security experts view any security requirements as excessive, or whether the nature of security unknowns inhibits experts from reaching this conclusion.

The MQM employs vignette surveys to link requirements as factors to a system quality, and to elicit expert judgements about quality levels achieved by those requirements. This is different from prior work in scenario-based requirements elicitation that employs interviews [14, 25, 32]. Although interviews provide detailed scenario descriptions, our approach allows analysts to attribute a quality level to specific requirements and their interactions. The MQM does not measure coverage, but it does increase coverage as the analyst explores more scenarios and collects new requirements to increase quality. In this paper, we evaluate the MQM as applied to security scenarios.

The inquiry cycle model uses scenarios while interviewing stakeholders [25], and the operational scenarios used by van Lamsweerde and Willemet are derived from interview excerpts and requirements documentation [15]. In MQM, we choose to use online surveys on security experts and generate scenarios using the factorial vignettes approach. By using factorial vignettes, we can define requirements of interest as factors, give each factor multiple descriptions which we call levels of the factor, and generate different combinations of levels that results in many different vignettes. The MQM offers increased coverage of scenarios as it allows the manipulation of descriptions, and the measurement effects of certain requirements on the outcome as well as the dependencies between the requirements. In addition, surveys make it more convenient to recruit more stakeholders, which increases the number of viewpoints of the scenario, and multiple viewpoints improve inter-personal uncertainty; which means, one expert might point out something that other experts missed while other experts find something different. This uncertainty among experts, which impacts security assessments [10, 11, 12], is due to differences in background or human memory limitations [11].

Acknowledgment

We thank our study participants and the SANS organizers. This research was supported by NSA Award #141333, and ONR Awards: #N00244-16-1-0006 and #NOO244-17-0012.

References

[1] D. Bates, M. Maechler, B. Bolker, S. Walker, R. H. B. Christensen, H. Singmann, and B. Dai, lme4: Linear mixed-effects models using Eigen and S4. 2014.

[2] J. McDermott and C. Fox, “Using abuse case models for security requirements analysis,” Comp. Security App. Conf., 1999.

[3] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, “G* Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences,” Behav. Res. Methods, vol. 39, no. 2, pp. 175–191, 2007.

[4] A. Fryrear, “What’s a Good Survey Response Rate?,” *SurveyGizmo*, 27-Jul-2015. [Online]. Available: https://www.surveygizmo.com/survey-blog/survey-response-rates/. [Accessed: 16-Feb-2017].

[5] S. Garfinkel, “Design principles and patterns for computer systems that are simultaneously secure and usable,” Massachusetts Institute of Technology, 2005.

[6] A. Gelman and J. Hill, Data analysis using regression and multilevel/hierarchical models. Cambridge Univ. Press, 2006.

[7] I. Graham, “Task scripts, use cases and scenarios in object oriented analysis,” Obj. Oriented Syst., vol. 3, no. 3, pp. 123–142, 1996.

[8] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, “Security requirements engineering: A framework for representation and analysis,” *IEEE Trans on Softw. Eng.*, vol. 34, no. 1, pp. 133–153, 2008.

[9] H. Hibshi, and T. D. Breaux, “Evaluation of Linguistic Labels Used in Applications,” Carnegie Mellon University, 2016.

[10] H. Hibshi, T. Breaux, and S. B. Broomell, “Assessment of Risk Perception in Security Requirements Composition,” 2015 IEEE 23rd Int. Requir. Eng. Conf. RE, pp. 146–155, Aug. 2015.

[11] H. Hibshi, T. D. Breaux, M. Riaz, and L. Williams, “A Grounded Analysis of Experts’ Decision-Making during Security Assessments,” *J. Cybersecurity*, 2016.

[12] H. Hibshi, T. D. Breaux, and C. Wagner, “Improving security requirements adequacy: an interval type 2 fuzzy logic security assessment system,” IEEE Symp. on Comp’l Intelligence, 2016.

[13] I. Jacobson, Object-oriented software engineering: a use case driven approach. Pearson Education India, 1993.

[14] A. Van Lamsweerde, “Requirements engineering in the year 00: a research perspective,” 22nd Int. Conf. on Soft. Eng., 2000, pp. 5–19.

[15] A. van Lamsweerde and L. Willemet, “Inferring declarative requirements specifications from operational scenarios,” IEEE Trans. Softw. Eng., vol. 24, no. 12, pp. 1089–1114, Dec. 1998.

[16] E. Letier, J. Kramer, J. Magee, and S. Uchitel, “Monitoring and control in scenario-based requirements analysis,” 28th Int. Conf. on Softw. Eng., 2005, pp. 382–391.

[17] Y. S. Lincoln and E. G. Guba, Naturalistic inquiry, vol. 75. Sage, 1985.

[18] L. Liu, E. Yu, and J. Mylopoulos, “Analyzing security requirements as relationships among strategic actors,” Symp. on Requir. Eng. for Info. S, 2002.

[19] D. Lüdecke, “sjPlot: data visualization for statistics in social science,” R Package Version, vol. 1, no. 4, 2015.

[20] H. Mouratidis and P. Giorgini, “Enhancing Secure Tropos to effectively deal with security requirements in the development of multiagent systems,” Safety and Security in Multiagent Sys., Springer, 2009, pp. 8–26.

[21] “NIST/ITL Special Publication (800),” 02-Jan-2015. [Online]. Available: http://www.itl.nist.gov/lab/specpubs/sp800.htm. [Accessed: 02-Jan-2015].

[22] B. Nuseibeh, J. Kramer, and A. Finkelstein, “A framework for expressing the relationships between multiple views in requirements specification,” IEEE Trans. Softw. Eng., vol. 20, no. 10, pp. 760–773, Oct. 1994.

[23] C. Potts, “Requirements models in context,” IEEE 3rd Int. Symp. on Requir. Eng., 1997, 1997, pp. 102–104.

[24] C. Potts and W. C. Newstetter, “Naturalistic inquiry and requirements engineering: reconciling their theoretical foundations,” IEEE 3rd Int. Symp. on Requir. Eng., 1997, pp. 118–128.

[25] C. Potts, K. Takahashi, and A. I. Anton, “Inquiry-based requirements analysis,” *IEEE Softw.*, vol. 11, no. 2, pp. 21–32, 1994.

[26] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013

[27] M. T. Roberson and E. Sundstrom, “Questionnaire design, return rates, and response favorableness in an employee attitude questionnaire.,” J. Appl. Psychol., vol. 75, no. 3, p. 354, 1990.

[28] P. H. Rossi and S. L. Nock, Measuring Social Judgments: The Factorial Survey Approach. SAGE Publications, 1982.

[29] J. Saldaña, The coding manual for qualitative researchers. Sage, 2012.

[30] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and quasi-experimental designs for generalized causal inference*. Houghton, Mifflin and Company, 2002.

[31] “SANS Institute: About.” [Online]. Available: https://www.sans.org/about/. [Accessed: 04-Feb-2017].

[32] A. Sutcliffe, “Scenario-based requirements analysis,” Requir. Eng., vol. 3, no. 1, pp. 48–65, 1998.

[33] L. Wallander, “25 years of factorial surveys in sociology: A review,” Soc. Sci. Res., vol. 38, no. 3, pp. 505–520, Sep. 2009.