Empir Software Eng (2017) 22:2684–2717
DOI 10.1007/s10664-016-9449-1

To log, or not to log: using heuristics to identify mandatory log events – a controlled experiment

Jason King[1] · **Jon Stallings**[2] · **Maria Riaz**[1] · **Laurie Williams**[1]

Published online: 24 August 2016

Abstract
Context User activity logs should capture evidence to help answer who, what, when, where, why, and how a security or privacy breach occurred. However, software engineers often implement logging mechanisms that inadequately record *mandatory log events* (MLEs), user activities that must be logged to enable forensics.
Goal *The objective of this study is to support security analysts in performing forensic analysis by evaluating the use of a heuristics-driven method for identifying mandatory log events.*
Method We conducted a controlled experiment with 103 computer science students enrolled in a graduate-level software security course. All subjects were first asked to identify MLEs described in a set of requirements statements during the pre-period task. In the post-period task, subjects were randomly assigned statements from one type of software artifact (traditional requirements, use-case-based requirements, or user manual), one readability score (simple or complex), and one method (standards-, resource-, or heuristics-driven). We evaluated subject performance using three metrics: statement classification correctness (values from 0 to 1),

Communicated by: Richard Paige, Jordi Cabot and Neil Ernst

✉ Jason King
jtking@ncsu.edu

Jon Stallings
jwstalli@ncsu.edu

Maria Riaz
mriaz@ncsu.edu

Laurie Williams
laurie_williams@ncsu.edu

[1] Department of Computer Science, North Carolina State University, 890 Oval Dr., Raleigh, NC 27695-8206, USA

[2] Department of Statistics, North Carolina State University, 2311 Stinson Dr., Raleigh, NC 27695-8203, USA

MLE identification correctness (values from 0 to 1), and response time (seconds). We test the effect of the three factors on the three metrics using generalized linear models.
Results Classification correctness for statements that *did not* contain MLEs increased 0.31 from pre- to post-period task. MLE identification correctness was inconsistent across treatment groups. For simple user manual statements, MLE identification correctness *decreased* 0.17 and 0.12 for the standards- and heuristics-driven methods, respectively. For simple traditional requirements statements, MLE identification correctness *increased* 0.16 and 0.17 for the standards- and heuristics-driven methods, respectively. Average response time decreased 41.7 s from the pre- to post-period task.
Conclusion We expected the performance of subjects using the heuristics-driven method to improve from pre- to post-task and to consistently demonstrate higher MLE identification correctness than the standards-driven and resource-driven methods across domains and readability levels. However, neither method consistently helped subjects more correctly identify MLEs at a statistically significant level. Our results indicate additional training and enforcement may be necessary to ensure subjects understand and consistently apply the assigned methods for identifying MLEs.

Keywords Logging · User activity logs · Security · Controlled experiment · User study · Mandatory log events

1 Introduction

A 2011 Veriphyr Survey of Patient Privacy Breaches (Survey of Patient Privacy Breaches 2011) in the healthcare domain revealed that the most commonly reported breaches of protected health information involved unauthorized access of medical records of others, including coworkers and relatives. In addition, 52 % of survey participants indicated that their organization did not have adequate tools for monitoring inappropriate access to sensitive patient data. Unauthorized access and disclosure of sensitive data occurs in many domains and affects multiple types of sensitive data, including tax records (McKenney 2014), driving records (Roper 2013), and education records (Read 2015). Unauthorized access of sensitive data can result in substantial civil and criminal penalties (Privacy 1974; Health Insurance Reform: Security 2013). Without adequate logging, detection of unauthorized access and forensic analysis may be impossible.

User activity logs should promote user accountability and enable forensic analysis by capturing evidence to help answer who, what, when, where, why, and how a security or privacy breach occurred (Chuvakin and Peterson 2010). However, software engineers often inadequately and inconsistently implement logging mechanisms that record "what" events must be logged for meaningful forensic analysis to hold users accountable (King et al. 2012a, b). Currently, specifications for logging mechanisms are documented in many different sources (King and Williams 2013), including industry standards, domain-specific guidelines, and certification criteria. Based on our findings (King and Williams 2013), no single centralized resource exists to guide software developers in implementing user activity logging mechanisms that consistently capture events needed to enable forensic analysis of user activity.

To help address the inadequacy and inconsistency of user activity logging mechanisms, in previous work (King et al. 2015), we proposed a systematic heuristics-driven method for identifying mandatory log events (MLEs) from statements contained in natural language

software artifacts. The method is driven by twelve empirically-derived heuristics. In our method, we express MLEs as <verb, object> tuples, where *verb* is the action the user performs and *object* is the resource being acted upon by the user. In this paper, we evaluate whether our heuristics-driven method actually improves a software engineer's ability to identify MLEs as compared with using existing industry standards.

The objective of this study is to support security analysts in performing forensic analysis by evaluating the use of a heuristics-driven method for identifying mandatory log events. We conduct a controlled experiment (Shull et al. 2008) to compare the ability of software developers to identify MLEs using three different methods:

- Standards-driven: participants are guided by real-world logging specifications from existing healthcare and Payment Card Industry logging standards.
- Resource-driven: participants are guided by first identifying any sensitive resources, then identifying the actions users perform when interacting with the resources.
- Heuristics-driven: participants are guided by our method (King et al. 2015) that uses twelve empirically-derived heuristics to determine which user actions should be logged.

Since most subjects have never seen or used the three methods before, we provide instructions on how to use the methods during the execution of the experiment. However, we do not have the ability to strictly enforce adherence to the methods or assess understanding of the methods during the execution of the experiment.

In addition to the three methods for identifying MLEs, we consider whether the software artifact type (traditional requirements, use-case based requirements, or user manual) or readability score (lower or higher) of the natural language software artifact may affect a software developer's ability to identify MLEs. For this study, 103 graduate-level computer science students enrolled in an introductory software security course at North Carolina State University (NCSU) serve as our subjects. We conduct the study online using the Qualtrics[1] online survey software.

Our research is guided by the following research questions:

RQ1: How does the *correctness of identifying whether a statement contains a mandatory log event* differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?
RQ2: How does the *correctness of the identified mandatory log events* differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?
RQ3: How does the *response time* of subjects identifying mandatory log events differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?

Our research contributes the following:

- An empirical evaluation and comparison of three methods for identifying MLEs;
- Materials for further experiment replication and analysis; and
- Lessons learned on how to improve both identification of MLEs and future experiment replication.

[1] http://www.qualtrics.com/

The remainder of this paper is organized as follows: Section 2 presents related work. Section 3 describes our heuristics-driven method for identifying MLEs. Section 4 describes our experiment planning. Section 5 discusses the execution of our experiment. Section 6 presents our data synthesis procedure. Section 7 presents our analysis methodology. Section 8 discusses our results. Section 9 presents our discussion of the results. Section 10 discusses lessons learned. Section 11 presents threats to validity. Section 12 summarizes and concludes the paper.

2 Related Work

Riaz et al. (Riaz et al. 2014a) empirically derived templates for documenting explicit security requirements for software systems using existing natural-language software artifacts. For accountability-related security requirements, any natural language sentence that contained a user acting upon a system resource implied the need for the software to log each time the user performs the action on the resource. In this work, we evaluate whether subjects in our experiment identify both the user action and the resource being acted upon in their descriptions of MLEs, or whether a subject identifies only an action or a resource. Riaz et al. (Riaz et al. 2014b) also performed a controlled experiment to evaluate the ability of software developers to document explicit security requirements, including logging requirements, using the templates derived earlier (Riaz et al. 2014a).

Yuan et al. (Yuan et al. 2012a) proposed an automated approach to improve the ability to diagnose faults through the use of logs for debugging. Similarly, the LogEnhancer (Yuan et al. 2012b) approach automatically suggests which variable values need to be recorded in each existing log message. Fu et al. (Fu et al. 2014) characterized logging practices and the use of logs in industry. However, all of these proposed approaches target only the fault diagnostic and debugging capabilities of logging. In contrast, our work seeks to advance user activity logging, specifically, as a means for enabling meaningful forensic analysis of user activity logs after a privacy or security breach.

Yskout et al. (Yskout et al. 2008) discussed a technique for transforming explicitly-stated logging requirements into an architectural model using Unified Modeling Language (UML). However, the approach only models explicitly-stated logging requirements, such as “The starting and stopping of the ATM machine needs to be audited.” In contrast, our work considers log events either explicitly-stated *or* implied by existing functional requirements. For example, the sentence “Doctors may edit prescriptions” describes the user action <edit, prescription>, which must be logged to enable user accountability and meaningful forensic analysis.

Vance et al. (Vance et al. 2013) discussed the importance of identifiability, the belief that one’s actions within a group can be associated with him or her individually. When individuals sense that they are distinguishable within a group, malicious behaviors tend to be curtailed. The researchers perform a factorial survey of 96 information systems students to investigate whether the awareness of logging of user actions influenced a person’s behavior. The researchers suggest auditing increases a user’s desire to engage in “approved” actions, thus decreasing intention to violate an access policy. Logging adequate traces of user activity enables auditing, which may deter users from violating an access policy because they know they could be held personally accountable for the actions performed.

3 Heuristics-Driven Method for Identifying Mandatory Log Events

In previous work, we proposed a systematic method for identifying MLEs from natural language software artifacts (King et al. 2015) in three domains: healthcare, human resources management, and academic conference management. In the study, two researchers collaborated to first identify all verb-object pairs contained in a traditional requirements specification from the human resource management domain, a use-case-based requirements specification from the healthcare domain, and a user manual from the conference management domain. The researchers individually classified each verb-object pair as *loggable* or *not loggable*. Based on discussions about disagreements in classifications, the authors empirically-derived a set of twelve domain-independent heuristics to help distinguish between actions that are loggable and those that are not loggable. Table 1 describes the twelve heuristics. Overall, the twelve heuristics helped facilitate classification of 96 % of the verb-object pairs contained in the study. From the previous study, we use the action-resource pairs classified as *loggable* as an oracle for our current controlled experiment.

4 Experiment Planning

In this section, we outline the goal and hypotheses of our experiment, and then describe our experimental design, experimental units, and materials.

4.1 Goal, Metrics, and Hypotheses

Using the Goal-Question-Metric approach (Solingen et al. 2002), we define the following research objective:

> To evaluate the use of our heuristics-driven method for identifying mandatory log events by software engineers in the context of software security.

Similarly, to answer our research questions, we state our null hypotheses as follows:

> **H_{01}**: *Statement classification correctness* is the same for all three methods used to identify MLEs.
> **H_{02}**: The *statement MLE identification correctness* is the same for all three methods used to identify MLEs.
> **H_{03}**: The *response time* of subjects on identifying MLEs for statements is the same for all three methods used to identify MLEs.

Table 2 summarizes the metrics used to address the research goals, questions, and hypotheses.

Statement classification correctness indicates whether the subject correctly classified a statement as containing MLEs or not. If the subject agreed with our oracle classifications, the value of the metric for the statement is 1. If the subject disagreed with our oracle, the value of the metric for the statement is 0.

Statement MLE identification correctness is, for a given statement, the proportion of unique subject responses that identified MLEs in our oracle relative to the total number of student responses, multiplied by the proportion of subject-identified MLEs in our oracle relative to the

Table 1 Heuristics for identifying mandatory log events

ID	Heuristic description	Example
HR1	*If the verb involves creating, reading, updating, or deleting resource data in the software system, then the event must be logged.*	**Statement:** A doctor creates a new prescription. **MLE:** <create, prescription> **Justification:** "creates" is one of *create, read, update, delete*
HR2	*If the verb can be accurately rephrased in terms of creating, reading, updating, or deleting resource data in the software system, then the event must be logged.*	**Statement:** A patient designates a personal representative. **MLE:** <designate, personal representative> **Justification:** "*designates*" *can be rephrased as* "*creates*"
HR3	*If the verb implies the system displaying or printing resource data that is capable of being viewed in the user interface or on paper, then the event must be logged.*	**Statement 1:** The system lists immunizations for a patient. **MLE:** <list, immunizations> **Statement 2:** The user sees a list of immunizations for a patient. **MLE:** <see, list of immunizations> **Statement 3:** A list of immunizations appears for the user. **MLE:** <appear, list of immunizations> **Justification:** Each statement involves resource data potentially being viewed
HR4	*If the verb expresses the intent to perform an action, such as "choose to", "select to", "plan to", or "wish to", then the intent event is not a mandatory log event.*	**Statement:** The user chooses to view a medication. **MLE:** <view, medication> **NOT an MLE:** <choose, to view a medication> **Justification:** "choosing" to do something is not an action that can be performed in the interface.
HR5	*If the verb expresses the granting or revocation of access privileges in the software system, then the event must be logged.*	**Statement:** The system shall allow doctors to edit medications. **MLE:** <allow, to edit medications> **Justification:** "allow" implies that doctors have access privileges to edit medications.
HR6	*If the verb is ambiguous, such as "provide" or "order", context must be considered when determining if the event must be logged.*	**Statement 1:** The system provides a list of immunizations for users. **MLE:** <provide, list of immunizations> **Statement 2:** The system provides a means for users to view immunizations. **MLE:** <view, immunizations> **Justification:** "provide" has multiple meanings in different contexts
HR7	*If the verb describes an action that takes place outside the scope of the functionality of the software, then the event is not a mandatory log event.*	**Statement:** To accept conference registration payments, setup a PayPal Business account. **NOT an MLE:** <setup, PayPal Business account> **Justification:** setting-up a PayPal Business account is performed outside the system
HR8	*If the verb involves the creation or termination of a user session, then the event must be logged.*	**Statement:** A user has successfully authenticated in the system. **MLE:** <authenticate, user> **Justification:** authentication involves the creation of a user session
HR9	*If the verb describes a state or quality within the system, then the event is not a mandatory log event.*	**Statement:** A patient is a registered user of the system. **NOT an MLE:** <is, registered user> **Justification:** "is a registered user" describes a *state* within the system

Table 1 (continued)

ID	Heuristic description	Example
HR10	*If the verb describes possession or composition of a resource or quality, then the event is not a mandatory log event.*	**Statement 1**: A patient has a known drug interaction. **NOT an MLE:** <has, a known drug interaction> **Statement 2:** The medication contains a dosage. **NOT an MLE:** <contain, dosage> **Justification:** "has" and "contain" describe possession and composition
HR11	*If the verb describes navigation or mechanical interaction with the software interface, then the event is not a mandatory log event.*	**Statement 1:** The doctor types the patient's name. **NOT an MLE:** <type, patient name> **Statement 2:** The user clicks submit. **NOT an MLE:** <click, submit> **Justification:** "type" and "click" bot describe mechanical interaction with the user interface
HR12	*If the verb describes initialization of the software or configuration of the software, then the event must be logged.*	**Statement:** The site administrator can install additional locales as they become available. **MLE:** <install, locale> **Justification:** installing new features involves reconfiguring the state of the software

total number of MLEs in our oracle. This metric looks at both the overall identification rate, but also measures efficiency by penalizing students for giving redundant responses about the same MLE.

For example, suppose a subject provides 4 responses for a statement, but only 2 of the responses actually describe unique MLEs contained in our oracle. The proportion of subject responses that correctly identify MLEs would be 0.5 (2 subject responses each identify unique MLEs contained in the oracle ÷ 4 total subject responses). If our oracle contains 3 MLEs for the statement, then the proportion of MLEs identified by the subject would be 0.66 (2 MLEs identified by the student ÷ a total of 3 MLEs in the oracle for the statement). The *statement MLE identification correctness* for the statement would, therefore, be 0.33 (0.5 × 0.66). In contrast, if the 2 subject responses both identified the *same* MLE in our oracle, then the proportion of subject responses that correctly identify MLEs would be 0.25 (1 subject response identifies a unique MLE contained in our oracle ÷ 4 total subject responses). In this case, with duplicate subject responses for the same MLE, the *statement MLE identification correctness* for the statement would, therefore, be 0.165 (0.25 × 0.66).

The *statement MLE identification correctness* metric calculation penalizes subjects for over-specification of MLEs, or for repeating the same MLE multiple times. The value of the metric will equal 1 in only two cases: (1) if the subject responses for the statement *exactly* identifies the set of MLEs in our oracle, with no omissions or extraneous responses; and (2) if both the subject and oracle agree that no MLEs are contained in the statement. We are primarily interested in *statement MLE identification correctness* for statements that *do* contain MLEs, based on our oracle.

Statement response time is the number of seconds a subject spent determining whether a statement contained MLEs, and if so, listing the MLEs for the statement. In Qualtrics, we use *page submit time* as the value of our response time metric. For each statement, we provide a limited amount of time (180 s) for subjects to identify whether the statement contains an MLE and, if so, to then identify each MLE.

Table 2 Summary of metrics for evaluating subject responses

Metric	Description	Type	Metric calculation			
Statement classification correctness	Indicates that the subject correctly (1) or incorrectly (0) identified whether the statement contained a MLE	Quantitative			Subject classification	
					Statement Contains MLEs	*Statement Does not contain MLEs*
			Oracle classification	*Statement Contains MLEs*	1	0
				Statement Does not contain MLEs	0	1
Statement MLE identification correctness	Indicates the degree to which the subject responses exactly matched our oracle, with a higher value meaning the subject responses more correctly matched our oracle	Quantitative			Subject classification	
					Statement Contains MLEs	*Statement Does not contain MLEs*
			Oracle classification	*Statement Contains MLEs*	(# unique subject responses that describe MLEs in the oracle)/(# total subject responses) × (# oracle MLEs identified by the student)/(# total MLEs in the oracle)	0
				Statement Does not contain MLEs	0	1
Statement response Time	Indicates the length of time (in seconds) the subject spent determining whether the statement contained MLEs, and if so, listing the MLEs for the statement	Quantitative	the number of seconds elapsed before the subject submits a response for a given statement			

4.2 Experimental Design

For this experiment, we used a within-subject factorial design, taking baseline measurements (during the pre-period task) on each subject, and then taking the same measurements post-treatment (during the post-period task). A within-subject design allowed us to compare each subject's measurements to himself or herself and account for significant variability between the subjects. We were primarily interested in finding significant differences in the responses between the pre- and post-period tasks, and determining whether those differences depended on the treatment.

After the pre-period task, subjects were randomly assigned to one of 18 treatment groups corresponding to the three factors: type of software artifact in which statements were contained (3 levels), readability of statements (2 levels), and method used to identify MLEs (3 levels). The factors are described in detail in Section 4(D).

The format of the experiment involved a pre-survey, orientation task, pre-period task, post-period (treatment) task, and post-survey. An overview of the experimental tasks appears in Fig. 1. A complete copy of all experimental materials is available on the study website.[2]

4.3 Pre-Survey

In the pre-survey, subjects were first presented with sixteen questions asking how many years of academic and professional experience they have in computer science, software engineering, software security, human resources software development, healthcare software development, conference management software development, natural language processing, and software logging. The pre-survey experience-related responses were self-reported by each subject. We separated each category into "academic experience" and "professional experience" with the intention that subjects would consider school as academic experience, but consider coop, internship, or work in industry as professional experience. In addition, subjects were then presented with eight statements to rate (on a Likert-like scale of "Very Poor" to "Very Good") their own ability to comprehend different aspects of English language and grammar, including the ability to understand spoken English, to identify parts-of-speech of words, and to use correct English grammar. Subjects are also asked how well they understood a 10-min video lecture and 5-question multiple choice quiz on logging that was assigned as homework for the previous night, as well as how well they understood the role of logging mechanisms within the field of software security.

4.4 Experiment Tasks

Each subject completed an orientation task, pre-period task, and post-period task. Each task consisted of one instruction page at the start of the task, followed by one statement page (see Section 5.B) for each statement assigned to the given task.

[2] http://go.ncsu.edu/loggingexperiment

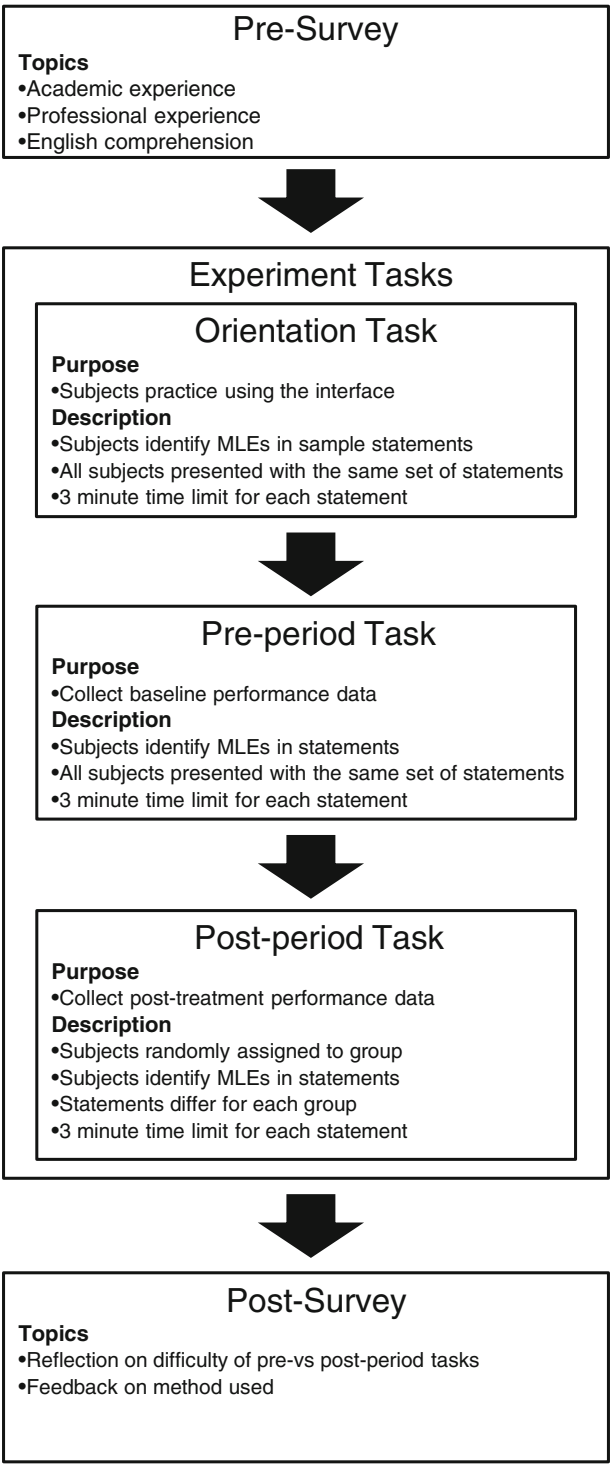


Fig. 1 Overview of experimental tasks

Orientation Task The purpose of the orientation task was to help subjects become familiar with the system interface used to conduct the experiment. Since we used the Qualtrics survey system to administer the entire experiment, we included a brief orientation task to familiarize subjects with the layout of the Qualtrics interface. The orientation task presented the same set of instructions, time limitations, page layouts, and input fields as the pre- and post-period tasks. Students were not evaluated on their ability to identify MLEs in the orientation task.

Pre-period Task The purpose of the pre-period task was to obtain a baseline assessment of a subject's ability to identify MLEs before the subject was randomly-assigned to a treatment group for the post-period task. During the pre-period task, in which all subjects were provided the same set of statements from which to identify MLEs.

Post-period Task The purpose of the post-period task was to obtain post-treatment assessment of subject performance to help determine whether subject performance changed from pre-period to post-period, and if subject performance differed among treatment groups. During the post-period task, students were provided statements based upon their randomly-assigned treatment group.

4.5 Post-survey

After completing the treatment activity, subjects were presented with a post-survey. In the post-survey, we asked four restrospective questions related to the perceived difficulty of the pre-period task versus the post-period task. We also asked subjects to briefly describe the method they used when identifying MLEs. Finally, we asked for any additional comments or feedback about the study or study tasks. For subjects assigned to the heuristics-driven method treatment, we also requested feedback regarding the heuristics.

4.6 Experiental Units

Subjects in our experiment are computer science students enrolled in a 15-week graduate-level software security course at North Carolina State University. Of the students enrolled in the course, 90 % were students enrolled in a Masters degree program. We performed the experiment during the 12th week of the course. We required subjects to perform the tasks in the experiment during a 75 min class period as course participation credit, but subjects could opt-out of allowing their responses to be included for the purpose of this research.[3] Of the 110 students enrolled in the course, 103 students consented to participate in the study. Section 5(B) further discusses how students prepared for the experiment.

4.7 Experiment Materials

In our experiment, we control three factors, summarized in Table 3:

- *Type of the natural language software artifact* from which subjects are asked to identify MLEs. Since our heuristics-driven method involves identifying verbs and objects from

[3] This study was approved by the North Carolina State University Institutional Review Board (#5354)

Table 3 Summary of experimental factors and levels

Factor	Levels
Type of natural language software artifact	Traditional requirements specification
	Use-case based requirements specification
	User manual
Readability of statements	Simple
	Complex
Method used to identify MLEs	Standards-driven
	Resource-driven
	Heuristics-driven

natural language statements, we evaluate whether the type of software artifact affects MLE identification.

- *Readability* of the set of statements contained in the natural language software artifact from which subjects are asked to identify MLEs. Since our heuristics-driven method involves identifying verbs and objects from natural language statements, we evaluate whether the readability of software artifact affects MLE identification.
- *Method used to identify MLEs.* We evaluate whether a heuristics-driven method improves MLE identification compared to two alternative methods for identifying MLEs: standards-driven and resource-driven methods.

4.7.1 Type of the Natural Language Software Artifact

To select the software artifact types to use in our experiment, we use the software artifacts contained in our previously-created oracle (King et al. 2015) of MLEs in human resources, healthcare, and conference management software artifacts:

- **iHRIS**[4] **v4.2:** Open Source Human Resources Information Solutions.

  – Software Artifact: traditional requirements specification.
  – According to the iHRIS community, 15 countries are using the software, with more than 675,000 health worker records currently supported in iHRIS.

- **iTrust**[5]**v18:** Open Source Electronic Health Record System.

  – Software Artifact: use-case-base requirements specification
  – An electronic health record (EHR) system developed and maintained by undergraduate software engineering students at North Carolina State University and used by many researchers and educators as a test-bed (Meneely et al. 2012).

[4] http://www.ihris.org/
[5] http://agile.csc.ncsu.edu/iTrust
[6] https://pkp.sfu.ca/ocs/

- **OCS (Open Conference Systems)[6]v2.3.6:** Open Source Scholarly Conference Management System.

  – Software Artifact: user manual
  – A conference management system developed and maintained by the Public Knowledge Project (PKP), a multi-university initiative developing free open-source software and conducting research to improve quality of scholarly publishing.

4.7.2 Statement Selection Based on Readability

To select statements to include in our experiment based on readability scores, we first split the natural language software artifacts into individual sections so that the statements in each section all describe related functionality. For the iHRIS traditional requirements specification, statements were hierarchically organized by topic. We sectioned the iHRIS statements based on the existing hierarchical organization topics. For the iTrust use-case based requirements specification, we sectioned the statements by individual use case. For the OCS user manual, we sectioned statements by first-order headings.

Next, we calculated[7] readability metrics for each section of statements in each source software artifact. Since readability metric algorithms measure different features of written language (for example, syllables per word, words per sentence, ignoring compound words, etc.), instead of relying on a single readability metric, we used scores from three readability metrics: Fleish-Kincaid (Kincaid et al. 1975), Automated Readability Index (ARI) (Smith and Senter 1967), and SMOG grade (McLaughlin 1969). For our experiment, we calculated Fleish-Kincaid, ARI, and SMOG scores for each candidate section of statements from each source artifact. Next, we sorted the scores in ascending order from simplest to most complex. For our experiment, we selected a section (or set of sections) with overall simple readability, and a section (or set of sections) with overall complex readability. To ensure consistency in the number of statements selected for each software artifact type and readability, we select multiple sections for TraditionalRequirements-Simple, TraditionalRequirements-Complex, UseCase-Simple, UseCase-Complex, UserManual-Simple, and UserManual-Complex. For a baseline set of statements, we select a set of statements from the iTrust use-case-based requirements specification since the iTrust specification contained the most sections available from which to select.

Because of differences in the number and grammatical style of statements available from which to select, we were unable to select sections with *identical* readability scores across each of the three types of software artifacts. For example, readability scores of traditional requirements statements ranged from 76.8 to 89.1 for Fleish-Kinkaid; 2.5 to 4.7 for ARI; and 2.3 to 8.5 for SMOG. Table 4 summarizes the software artifact statements used in our experiment.

4.7.3 Method Selection

To select the different methods subjects use to identify MLEs, we begin by including the heuristics-driven method at the center of our experiment. Next, we define a standards-driven

[6] https://pkp.sfu.ca/ocs/
[7] To calculate readability metric values, we use the calculators provided by https://readability-score.com/

Table 4 Summary of software artifact statements for each software artifact type and readability

Artifact type	Readability	Number of statements	Flesch-Kinkaid reading ease (0–100, where 100 is simple)	Automated readability index (Grade level, where 0 is simple)	SMOG grade (Grade level, where 0 is simple)
Baseline (from a Use-case-based Requirements Specification)	Mid-range	17	73.1	12.1	11.6
Traditional requirements specification	Simple	14	89.1	2.5	2.3
	Complex	13	76.8	4.7	8.5
Use-case-based requirements specification	Simple	23	77.4	1.9	4.8
	Complex	20	44.5	13.9	13.7
User manual	Simple	13	64.5	7.7	8.3
	Complex	16	33.7	13.4	13.5

method that reflects real-world software development by providing only actual industry standards from healthcare (Standard Specification for Audit and Disclosure Logs for Use in Health Information Systems 2013) and the Payment Card Industry (Payment Card Industry Data Security 2010) as guidance for subjects. Finally, since our heuristics-driven method focuses on identification of *verbs*, we define a resource-driven method that involves first identifying *resources* being acted upon, then identifying the actions performed against the resource. Neither the standards-driven or resource-driven methods provide heuristics for guidance. The full set of method descriptions can be found on the study website[8] and in Table 5.

5 Experiment Execution

In this section, we discuss subject preparation and experiment execution procedure.

5.1 Preparation

Before participating in the experiment, subjects were required to watch a 10-min video lecture on software logging in the context of software security. The video lecture did not reveal any techniques for identifying MLEs. Instead, the video provided overall descriptions of why logging is important for nonrepudiation, and outlined common weaknesses and vulnerabilities with logging mechanisms. Subjects also completed a 5-question multiple-choice quiz on logging that was assigned as homework for the previous night.

[8] http://go.ncsu.edu/loggingexperiment

Table 5 Summary of methods and instructions provided for each

Method	Instructions provided
Standards-driven	**Industry Standard 1:** An audit log is a record of actions (queries, views, additions, deletions, changes) performed on data by users. Actions should be recorded at the time they occur. These actions include user authentication, user or system-directed signoff, data access to view, and receipt of data content from external provider/practitioner. **Industry Standard 2:** The software must create a secure audit record each time a user: • accesses, creates or updates sensitive data via the software; • overrides the consent directives of a user via the software; • accesses, via the software, data that is locked or masked by instruction of a user; or • accesses, creates or updates registration data on a software user. **Industry Standard 3:** Implement audit trails to link all access to system components to each individual user. Implement automated audit trails for all system components to reconstruct the following events: • all individual user accesses to data • all actions taken by individuals with root or administrative privileges • access to all audit trails • invalid logical access attempts • use of and changes to identification and authentication mechanisms -- including, but not limited to creation of new accounts and elevation of privileges -- and all changes, additions, or deletions to accounts with root or administrative privileges • initialization, stopping, or pausing of the audit logs • creation and deletion of system level objects
Resource-driven	**Technique:** In grammar, nouns are the fundamental construct that express objects, while verbs generally express actions being executed against an object. To identify user activity in natural language software artifacts: 1. First identify all nouns in the given sentence. For example, "Doctors prescribe medications for patients." Three nouns appear in this sentence: doctors, medications, and patients. 2. Then identify any verbs that act upon the nouns. In the sentence "Doctors prescribe medications for patients," no verbs act upon the nouns "doctors" or "patients". However, the verb "prescribe" describes an action performed on medication data. 3. Finally, determine if the action taken upon the object is loggable. Since neither the nouns "doctors" or "patients" are acted upon by a verb, neither are loggable. However, "prescribe medications" describes the creation of data in the system and may be considered loggable.
Resource-driven	**Technique:** In grammar, nouns are the fundamental construct that express objects, while verbs generally express actions being executed against an object. To identify user activity in natural language software artifacts: 1. First identify all nouns in the given sentence. For example, "Doctors prescribe medications for patients." Three nouns appear in this sentence: doctors, medications, and patients. 2. Then identify any verbs that act upon the nouns. In the sentence "Doctors prescribe medications for patients," no verbs act upon the nouns "doctors" or "patients". However, the verb "prescribe" describes an action performed on medication data. 3. Finally, determine if the action taken upon the object is loggable. Since neither the nouns "doctors" or "patients" are acted upon by a verb, neither are loggable. However, "prescribe medications" describes the creation of data in the system and may be considered loggable.
Heuristics-driven	**Technique:** In grammar, verbs are the fundamental construct that express an action being executed against an entity (indicated by an object). We express a verb-object pair as a tuple of the form < verb, object>. To extract verb-object pairs, we consider the following guidelines for each statement: • Explicitly stated verb-object pairs. Extract any verbs contained in the sentence, then identify any objects being acted upon by the verb ○ Example 1: "Doctors prescribe medications." ○ verb-object pair: <prescribe, medication> • Implied verb-object pairs. Extract any words in the sentence whose lemma is a verbal (e.g., gerunds, participles, and infinitives are verbals that function as nouns in a sentence), then identify any objects being acted upon by the verbal ○ Example 2: "Creating a patient…"

Table 5 (continued)

Method	Instructions provided
	○ verb-object pair: <create, patient> ○ Example 3: "The submitted proposal…" ○ verb-object pair: <submit, proposal> • Compound verb-object pairs. For any sentence that contains compound verbs or more than one object for a single verb, we document multiple verb-object pairs to consider each individual combination of verb and object: ○ Example 4: "Doctors prescribe and update medications" ○ verb-object pair: <prescribe, medication> ○ verb-object pair: <update, medication> Once all verb-object pairs are identified for a given statement, consider the following set of heuristics to help determine if the verb-object pair should be logged or not[a].

[a] the set of heuristics appears in Table 1 in this paper

5.2 Procedure

At the beginning of the experiment, subjects were informed they were to complete a set of tasks for classroom participation credit. However, subjects were also informed of the option to consent to the inclusion of their task responses for the purpose of this study. Subjects were provided a URL to the experiment website, hosted by Qualtrics. Once the URL was provided to subjects, they were given until the end of the 75 min class period to complete their responses to the study. Data collection and randomization of subjects into individual artifact type/readability/method groups were managed by the Qualtrics system. To ensure confidentiality, subjects were physically handed a random six-digit access code on a slip of paper at the beginning of the class period. During the study, subjects were asked to enter the random six-digit access code, which ensured that personally identifiable information was never collected.

For each task, subjects were presented with two types of pages. First, each task began by presenting a page of instructions. After the instructions page, subjects were presented one page for each statement in the set of statements for the given task.

5.2.1 Instruction Page

For the orientation, pre-period, and post-period tasks, we first presented instructions for how to respond to the questions during the task. The instructions provided links to any additional reference materials. For example, during the post-period task, the instruction page described the method for identifying MLEs that was assigned to the subject. Subjects could spend as much time as necessary reading the instructions and any reference materials, with the understanding that they complete the entire experiment within the overall 75-min class period. The instructions also informed subjects of a 3 min time limit for providing a response on each statement page before being auto-advanced to the next statement. We imposed a 3 min time limit for each statement to keep all subjects on pace to complete the task within the 75-min time allotted for the experiment.

5.2.2 Statement Pages

After the instruction page, subjects were presented with one page for each statement in the set of statements assigned for the given task. Figure 2 provides ans example statement page. The statement page presented the text of the current statement and a question asking if the subject thought any MLEs were described in the statement. If the subject indicated that the statement described a MLE, the subject was then presented with a table of text fields for identifying up to ten MLEs and brief justifications for why the subject thought each MLE should be logged. In addition, the statement page displayed a countdown timer showing how much time was remaining before the page automatically submitted and advanced to the next statement.

For example, one statement presented to students contained the text, "A patient or personal health representative or LHCP chooses to view prescription reports." Students assigned to use the standards-driven method for identifying MLEs read the instructions (see Table 5) for the standards-driven method. Similarly, students assigned to use the resource-driven method for identifying MLEs read the instructions (see Table 5) for the resource-driven method. Students assigned to use the heuristics-driven method read the instructions (see Table 5) for the heuristics-driven method, in addition to the 12 heuristics (see Table 1) for identifying MLEs. Based solely on the information presented in the instructions, the subject decided whether the statement described MLEs or not. If so, the subject described what they thought the MLE was in the text fields provided. Figure 2 presents an example.

6 Synthesizing Raw Response Data

In this section, we describe our process for transforming the raw subject response data into the quantitative metrics used for testing the research hypotheses. Each subject response consists of

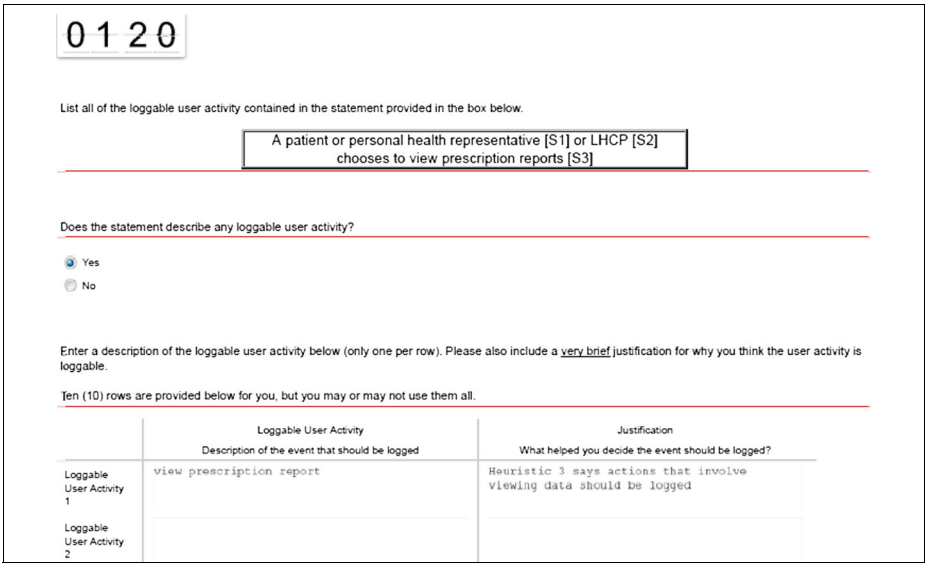


Fig. 2 Example statement page containing a countdown timer, options for selecting whether the statement describes any MLEs, and (if so), text fields for identifying up to 10 MLEs

two main elements: the source statement text, and the list of MLEs identified by the subject for the given source statement. While synthesizing the raw response data, we removed 2 subjects from the study because they copied-and-pasted the source statement as their response for MLE identification. We include a total of 101 subjects in our experiment.

6.1 Data Management

After exporting raw response data from the Qualtrics survey system, we created an individual Excel workbook to manage data for each individual subject response. Each workbook contained two sheets: a sheet for managing pre-period task response data, and a sheet for managing post-period task response data. On each data sheet, we listed each of the source software artifact statements assigned for the given task. For each source software artifact statement, we included two columns:

- *MLEs identified by the subject* for the given statement, and
- *MLEs appearing in our oracle* (King et al. 2015) for the given statement. Our full oracle is provided on our study website[9]

6.2 Classifying Subject Responses

The first and third authors individually classified each subject response as one of the following:

- *Appears as loggable in the oracle*: the response correctly identified a verb-object pair classified as an MLE in our oracle
- *Appears as not loggable in the oracle*: the response identified a verb-object pair classified as not an MLE in our oracle
- *Not in the oracle, but loggable*: The subject identified an MLE that is not contained in our oracle
- *Not in the oracle, and not loggable*: The subject identified an action that is not contained in our oracle, but the identified action was not an MLE.
- *Not a description of an activity*: The subject provided responses that did not describe actions. For example, "username" and "timestamp" are data elements that should be logged for each log entry, but they *do not* represent an action that must be logged.
- *Activity is not described in the given* statement: The subject provided a description of an activity that was neither explicitly nor implicitly described in the provided statement. For example, "login user" is not described in the statement "A user views data."

For each subject response classified as *Appearing as loggable in oracle* or *Not in the oracle but loggable*, we further classified the completeness of the response as one of the following:

- *Identified both the action and the resource being acted upon* (for example: "view prescription data")
- *Identified only the action being performed* (for example: "view data")
- *Identified only the resource being acted upon* (for example: "prescription data")

[9] http://go.ncsu.edu/loggingexperiment

We also individually classified each of the MLE items in the oracle column as one of:

- *Identified by the subject:* at least one subject response described the given MLE
- *Not identified by the subject*: no subject response described the given MLE

Since subject responses could widely vary from the exact MLE text contained in our oracle, each rater considered synonymy and polysemy when classifying subject responses. For example, a student response of "see prescription report" was considered equivalent to our oracle's MLE description of "view prescription report." By having two different researchers classifying subject responses, we helped ensure consistent classification. In the event the two researchers disagreed on a classification, the disagreements were discussed and resolved.

6.3 Resolving Classification Disagreements

After individually classifying each subject response and each oracle MLE, we computed inter-rater agreement using the Cohen's Kappa metric. A larger k coefficient is considered an indicator of higher inter-rater agreement (Carletta 1996). Our overall Cohen's Kappa score for subject response classifications was $k = 0.70$, for completeness-of-response classifications was $k = 0.77$, and for oracle event classifications was $k = 0.78$. A breakdown of additional Cohen's Kappa values for each pre-period and post-period task are available on our project website.

For disagreements in classifications, the two researchers met to justify their decisions. After discussing and resolving all disagreements, we produced a final set of classifications for each subject.

6.4 Metric Calculations

For each statement in the pre- and post-period tasks, we calculated the value of *statement classification correctness*, *statement MLE identification correctness*, and *statement response time* based on the formulae provided in Table 2. The *statement classification correctness* and *statement MLE identification correctness* metrics are calculated based on our classifications of subject responses as described in Section 6.B.

In a spreadsheet, we recorded the metric values for each statement for each subject who participated in the study. Each row of the spreadsheet contained the following columns of data:

- Subject ID: the unique subject identifier.
- Task: the task in which the statement appeared: *pre-period*, or *post-period*.
- Statement Type: the type of artifact the statement was contained in: a *traditional requirements specification*; *use-case based requirements specification*; or *user manual*.
- Readability: the readability category of the set of statements the statement was contained: *baseline* for pre-period task statements; *simple readability* or *complex readability* for post-period task statements.
- Method: the method randomly-assigned for the subject to use when identifying MLEs: *standards-driven*; *resource-driven*; or *heuristics-driven*.
- Statement ID: the unique identifier for the given statement
- Has MLEs in the Oracle: 0, if the oracle does not contain any MLEs for the given statement; 1, if the oracle does contain MLES for the given statement.

- Statement Classification Correctness: the value of the statement classification correctness metric for the given statement.
- Statement MLE Identification Correctness: the value of the statement MLE identification correctness metric for the given statement.
- Statement Response time: the value of the statement response time metric for the given statement.

7 Analysis Methodology

We first describe our procedure for analyzing statement classification correctness. Next, we describe our procedure for analyzing statement MLE identification correctness. Finally, we describe our procedure for analyzing statement response time. All analyses were performed using either SAS 9.4 or JMP[10] Version 12.

7.1 Analyzing Statement Classification Correctness

To address H_{01}, we determine whether any of the three methods for identifying MLEs significantly changed the subject's rate of detection of loggable or non-loggable events between the pre- and post-period tasks. Therefore, for each subject, we averaged *statement classification correctness* across the four categories:

- Pre-period task statements that contain MLEs in the oracle
- Pre-period task statements that do not contain MLEs in the oracle
- Post-period task statements that contain MLEs in the oracle
- Post-period task statements that do not contain MLEs in the oracle

We fit a binomial generalized linear mixed model, which was preferred over a linear mixed model because of the large number of perfect classifications (where the average classification correctness equaled 1). This model was also able to incorporate different variances for groups due to an unequal number of statements, which was especially true for the three groups UserManual-Simple, TraditionalRequirements-Simple, and TraditionalRequirements-Complex.

Since we are performing an exploratory study to determine which effects *may* affect a subject's ability to correctly classify a statement as containing MLEs, we consider any effect with a p-value less than 0.1 to be a *potentially* significant effect for further exploration of the effect using pairwise contrasts.

The fixed effects of the model included all main effects and interactions for five factors: task period (pre- and post-), software artifact type, readability, method used to identify MLEs, and whether the statement contained MLEs in the oracle (yes or no). Random effects were included for each subject. We were uninterested in main effects for software artifact type, readability, and method, since these effects are averaged across the task periods and differences should only be significant during the post-period. Therefore, the primary focus of our analysis was to find significant interactions involving task period.

[10] http://www.jmp.com

For the interaction effects found to be significant, we investigated the change in the statement classification correctness between the pre- to post-period tasks for each level of the other factors involved in the interaction. For example, if a significant interaction was found between task period and artifact type, we looked at this difference for all three levels of artifact type. *T*-tests with Tukey adjustments were used to test if these changes were significantly different from 0. Changes were deemed significant if the adjusted p-values were less than 0.01.

7.2 Analyzing Statement MLE Identification Correctness

To address H_{02}, we determine whether students can correctly identify the action and resource components of the MLE, itself. Therefore, we look at *statement MLE identification correctness*. For each subject in both the pre- and post-period tasks, we averaged *statement MLE identification correctness* for only the statements that contain MLEs in the oracle. We fit a linear mixed model to this data with the same fixed and random effects as we did in Section 7(A). While *statement MLE identification correctness* ranges from 0 to 1, 91 % of subjects had an average MLE identification correctness greater than 0 and less than 1. Therefore, the linear mixed model is an appropriate approximation. A normal QQ-plot of the studentized residuals was used to verify the normality assumption.

The number of loggable statements was comparable across all the artifact-type/readability groups, so we did not need to account for unequal variances due to differing numbers of loggable statements. Since, at this stage of analysis, we are performing an exploratory study to determine which effects *may* affect a subject's ability to correctly identify MLEs for a statement, we consider any effect with a p-value less than 0.1 to be a *potentially* significant effect for further exploration of the effect using pairwise contrasts.

For the interaction effects found to be significant, we investigated the change in the proportion of MLEs identified between the pre- to post-period task for each setting of the other factors involved in the interaction. *T*-tests with Tukey adjustments were used to test if these changes were significantly different from 0. Changes were deemed significant if the adjusted p-values were less than 0.01.

7.3 Analyzing Statement Response Time

To address H_{03}, we compare the mean page submit times for each subject. For each statement, students had up to 180 s (3 min) to determine whether a statement contained MLEs and, if so, to list the MLEs. For each subject, we averaged *response time* across the four categories:

- Pre-period task statements that contain MLEs in the oracle
- Pre-period task statements that do not contain MLEs in the oracle
- Post-period task statements that contain MLEs in the oracle
- Post-period task statements that do not contain MLEs in the oracle

We calculated the mean response time and standard deviation for each of the four groups of values above. Since the standard deviations were unequal across the four groups of values, we fit the data using a weighted linear mixed model with weights equal to the inverse standard deviation of response times for each participant in the pre- and post-period tasks.

The fixed effects of the model for response time included all main effects and interactions for five factors: task period (pre- and post-), software artifact type, readability, method used to

identify MLEs, and whether the statement contained MLEs in the oracle (yes or no). Random effects were included for each subject. Just like in 7(A), the primary focus of our analysis was to find significant interactions involving task period.

For the interaction effects found to be significant, we investigated the change in the response time between the pre- to post-period tasks for each level of the other factors involved in the interaction. *T*-tests with Tukey adjustments were used to test if these changes were significantly different from 0. Changes were deemed significant if the adjusted p-values were less than 0.01.

8 Results

In this section, we first summarize subject pre-survey responses. In addition, we answer our research questions regarding the statement classification correctness, statement MLE identification correctness, and response time of identifying MLEs. Table 6 summarizes the number of subjects in each treatment group during the post-period task.

8.1 Subject Experience

We computed the mean, median, and standard deviations of all metrics for academic and professional experience and English comprehension of subjects (see Section 3-C) for all combinations of the three factors (software artifact type, readability, and method used to identify MLEs), and found the values to be similar across all groups. The overall average for academic performance in computer science, software engineering, and software security was 5, 2.5, and 1 year, respectively. Subjects had minimal experience in human resources, healthcare, and conference management software development, with averages of 0.2 years. Similar outcomes were observed with professional experience, but the averages tended to be lower. For example, the overall average for professional experience in computer science was 2 years. The consistency in scores across groups implies that initial ability of subjects is not a

Table 6 Summary of the number of subjects in each treatment group

ArtifactType	Readability	Method for identifying MLEs			Total
		Standards-driven	Resource-driven	Heuristics-driven	
Traditional requirements	Simple	6	6	6	18
	Complex	5	6	6	17
UseCase Based requirements	Simple	6	4	6	16
	Complex	6	6	6	18
User manual	Simple	6	6	5	17
	Complex	5	6	4	15
Total		34	34	33	101[a]

[a] 2 subjects were removed because they copy-and-pasted the source statement as their response, without actually identifying any MLEs

confounding factor. For example, no one group is significantly better at reading comprehension than another.

The pre-survey metrics were initially included in the task metrics analysis models, but none revealed any meaningful, significant relationships. Any significant relationship that was found with the procedure was due to outliers of the pre-survey metrics. For example, one subject claimed their academic experience in computer science was 15 years. Therefore, we did not consider the pre-survey metrics as potential covariates in the analysis of the task metrics.

8.2 Statement Classification Correctness

When fitting a factorial model, a widely-accepted practice is to remove insignificant interactions that involve a large number of factors. Removing insignificant interactions led us to a model that included all main effects and two-factor interactions. Fixed effect tests of all potentially significant effects are shown in Table 7. The full table of fixed effect tests can be found on our study website.[11]

The most highly significant effects only involved task period and whether the statement contained an MLE in the oracle. The interaction between task period and whether the statement contained an MLE in the oracle was the strongest effect. Classification correctness averages, as well as their differences, for all four of the combinations appear in Table 8.

Since we investigated any effect with a p-value less than 0.10, other possibly significant effects were the interactions Task*ArtifactType and Task*Readability, the latter of which was expected since readability only played a role in the post-period task. Classification correctness averages, as well as their differences, for Task*ArtifactType and Task*Readability are shown in Tables 9 and 10, respectively.

> **H_{01}**: *Statement classification correctness* is the same for all methods used to identify MLEs.

We fail to reject H_{01}. We observe no meaningful statistical difference in *statement classification correctness* for subjects using the standards-driven, resource-driven, and heuristics-driven methods.

8.3 Statement MLE Identification Correctness

The full factorial model was appropriate in this scenario. Fixed effect tests of all potentially significant effects are shown in Table 11.

For statement MLE identification correctness, we found a significant interaction with task*readability. However, the task*readability interaction does not involve the method used to identify MLEs, which is what we are interested in. We found a potentially significant four-way interaction between task period, software artifact type, readability, and method for identifying MLEs. The interaction suggests that the change from the pre- to post-period task might be affected by method, but the difference may depend on the artifactType/readability combination. Therefore, *t*-tests with Tukey-adjusted p-values were used to test the change in

[11] http://go.ncsu.edu/loggingexperiment

Table 7 Type III tests of fixed effects of potentially significant effects for statement classification correctness, sorted by Pr > F

Effect	Num DF	Den DF	*F* value	Pr > F
Task	1	290	65.08	<0.0001
OracleHasMLEs	1	290	66.92	<0.0001
Task*OracleHasMLEs	1	290	131.55	<0.0001
Task*Readability	1	290	9.88	0.0018
Task*ArtifactType	2	290	5.62	0.0040
Readability	1	87	5.06	0.0270
ArtifactType*OracheHasMLEs	2	290	3.42	0.0340

average MLE identification correctness from the pre- to post-period for all 18 treatment groups. The full table for the MLE identification correctness averages across all 18 artifactType*readability*method combinations can be found on our project website.[12] We did not observe any significant differences in any of the artifactType*readability*method groups.

> **H_{02}**: The *statement MLE identification correctness* is the same for all methods used to identify MLEs.

We fail to reject H_{02}. We do not observe any significant difference in *statement MLE identification correctness* among subjects assigned to standards-driven, resource-driven, and heuristics-driven methods for identifying MLEs.

8.4 Response Time of Identifying MLEs

The final factorial model included all effects up to three factor interactions. Fixed effect tests of all potentially significant effects are shown in Table 12.

Both the main effect tests for task period and OracleHasMLEs were highly significant and explained most of the variation in the response times. The three factor interaction between task period, software artifact type, and readability was also significant. We present the average response times, as well as their differences, in Table 13. All differences between the pre- and post-period task for all artifactType*readability groups were significant at the 0.05 level.

> **H_{03}**: The *response time* of subjects on identifying MLEs for statements is the same for all methods used to identify MLEs.

We fail to reject H_{03}. We observe no meaningful significant difference in response time between subjects who used the standards-driven, resource-driven, and heuristics-driven methods for identifying MLEs.

[12] http://go.ncsu.edu/loggingexperiment

Table 8 Classification correctness averages for Task*OracleHasMLEs

		Task period		Diff means
		Pre	Post	
OracleHasMLEs	Yes	0.62	0.55	−0.07**
	No	0.55	0.86	0.31**

** adjusted p-value < 0.01; * adjusted p-value < 0.05

9 Discussion

In this section, we answer our research questions.

9.1 Statement Classification Correctness

> **RQ1:** How does the *correctness of identifying whether a statement contains a mandatory log event* differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?

Even though no significant difference exists among the three different methods, *statement classification correctness* improved from the pre- to post-period tasks for statements that did not contain MLEs, for simple readability statements, and for the traditional requirements and use-case based requirements artifact types. Proportions of correct identification of statements that contain MLEs dropped from 0.62 to 0.55, which was significant with an adjusted *P*-value of 0.0026.

In Tables 9 and 10, we observe an increase in identification of whether statements do or do not contain MLEs, but the increase depends on both the artifact type and the readability of the statement. For example, the increase for simple statements was more dramatic than for complex statements (0.20 vs 0.10). The increases for TraditionalRequirements and UseCaseBasedRequirements and are similar (0.17 and 0.20, respectively) but the difference for UserManual is somewhat muted (0.07). However, we also observed that the UserManual group performed better in the pre-period than the TraditionalRequirements and UseCaseBasedRequirements groups, but the post-period task values for subjects assigned to UserManual statements were slightly worse.

Table 9 Classification correctness averages for Task*ArtifactType

		Task period		Diff means
		Pre	Post	
ArtifactType	TraditionalRequirements	0.59	0.76	0.17**
	UseCaseBasedRequirements	0.55	0.75	0.20**
	UserManual	0.63	0.70	0.07*

** adjusted p-value < 0.01; * adjusted p-value < 0.05

Table 10 Classification correctness averages for Task*Readability

		Task period		Diff means
		Pre	Post	
Readability	Simple	0.58	0.78	0.20**
	Complex	0.59	0.69	0.10**

** adjusted p-value < 0.01; * adjusted p-value < 0.05

No differences were found between the three methods for identifying statements that contain MLEs.

The correctness of identifying whether a statement contains a mandatory log event does not differ among subjects using the standards-driven, resource driven, and heuristics driven methods.

9.2 Statement MLE Identification Correctness

> **RQ2:** How does the *correctness of the identified mandatory log events* differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?

Even though Type III Tests of MLE identification correctness indicated a *possible* significant interaction between task, artifactType, readability, and method, further analysis did not reveal any significant differences between the three methods. The lack of significant differences suggests that the readability effect was strong enough to overcome any differences between the methods.

Lower values of *statement MLE identification correctness* could be due to a lack of identifying the action/resource of the MLE, over-specification of MLEs, or listing the same MLE multiple times for the same statement. Given that subjects were required to read and understand descriptions of the methods for identifying MLES on their own and in a relatively short period of time, subjects may have misunderstood or applied the methods incorrectly.

Statement MLE identification correctness did not significantly differ among subjects using the standards-, resource-, and heuristics-driven methods. Subjects may not have understood how to use the methods correctly within the time constraints.

Table 11 Type III tests of fixed effects of potentially significant effects for statement MLE identification correctness, sorted by Pr > F

Effect	Num DF	Den DF	F value	Pr > F
Task*Readability	1	83	9.37	0.0030
Task*ArtifactType*Readability*Method	4	83	3.64	0.0087

Table 12 Type III tests of fixed effects of potentially significant effects for response time, sorted by Pr > F

Effect	Num DF	Den DF	*F* value	Pr > F
Task	1	190.0	261.38	<0.0001
OracleHasMLEs	1	161.3	290.05	<0.0001
Task*ArtifactType*Readability	2	182.7	7.31	0.0009
Task*OracleHasMLEs*ArtifactType	2	161.7	2.84	0.0615
Readability	1	146.1	3.48	0.0642

9.3 Statement Response Time

> **RQ3:** How does the *response time* of subjects identifying mandatory log events differ among subjects using the standards-driven, resource-driven, and heuristics-driven methods?

From Table 12, the Task*ArtifactType*Readability interaction is significant ($p = 0.0009$). However, the interaction may only be significant because the pre-period response times for the subjects in the TraditionalRequirements-Complex group were higher than both the UseCaseBasedRequirements- and UserManual-Complex response times, yet the post-period response times for subjects in the TraditionalRequirements-Complex group were less than the UseCaseBasedRequirements- and UserManual-Complex post-period response times. For example, we found the response times for subjects in the TraditionalRequirements-Complex group dropped dramatically, from 84.69 s during the pre-period task, to 27.7 s during the post-period task. The overall decrease in response times in all treatment groups from the pre- to post-period task may be due to increased familiarity with the task of identifying MLEs or disinterest/pressure to finish the task before the end of the class period.

The three methods for identifying MLEs did not have significantly different response times. The lack of a significant difference in response times may be evidence that subjects assigned to the heuristics-driven method did not truly follow the instructions, since the heuristics-driven method included more reference materials for subjects to consider and should have taken longer than the standards- and resource-driven methods.

We did not observe any significant differences in response times between subjects who used the standards-driven, resource-driven, and heuristics-driven methods for identifying MLEs. This may, in part, be due to the time limit for responding to each statement.

10 Lessons Learned

When designing the experiment, we considered a number of practices to minimize biases and subjectivity. For example, we accounted for each subject's prior experience so that we can reliably compare the effects of the factors in the experiment. Providing a pre-period task helped in establishing a baseline of subject performance, while the pre-survey gathered self-reported data on subject experience. The post-survey solicited feedback from subjects related to the

Table 13 Response time averages (in seconds) for Task*ArtifactType*Readability

ArtifactType	Readability	Task period		Diff mean
		Pre	Post	
TraditionalRequirements	Simple	59.61	29.50	−30.11*
	Complex	84.69	27.70	−56.99*
UseCaseBasedRequirements	Simple	73.58	23.29	−50.29*
	Complex	64.62	30.74	−33.88*
UserManual	Simple	73.96	29.24	−44.72*
	Complex	73.02	38.84	−34.18*

** adjusted p-value < 0.01; * adjusted p-value < 0.05

method for identifying MLEs, as well as any observations related to the overall experiment design and execution. In this section, we provide an overview of the lessons learned as part of the current experiment.

10.1 Provide Both Verbal & Written Instructions

We noticed that a number of subjects seemingly misunderstood the instructions. Instead of identifying the MLEs in terms of actions performed by users (such as "view prescription"), some subjects focused on identifying the contents of the log entry (such as "timestamp" and "username"). While identifying log entry contents is an important aspect of accountability, identifying the actions that should trigger a log entry in the first place is necessary for adequately capturing user activity to support meaningful forensics. *To improve the subject's understanding of the task, explicit verbal instructions may be required to mitigate the risk of subjects ignoring or quickly scanning written instructions*.

10.2 Remove Time Constraints

According to post-survey responses, several subjects indicated that, in some cases, they required more than the maximum of 3 min allocated per statement to identify the MLEs as part of the pre- and post-period tasks. We observed that, for all statements in the study, subjects spent the full 180 s on 45 % of the statements presented. However, other subjects noted that working on one statement after the other can be tiring, and the ability to pause the task or to revisit responses to previous statements might be helpful. *In future experiments, one should consider allowing subjects to control the pace of the task while assigning an overall time limit to encourage task completeness*.

10.3 Provide More Training on Methods for Identifying MLEs

For the post-period task, after subjects were randomly assigned a treatment group, subjects were presented with an instruction page describing the method they were assigned for identifying MLEs. Even though we allowed subjects to remain on the instruction page as long as necessary to read and understand how to use the method, the value of our response time metric did not significantly differ for subjects using different methods for identifying

MLEs, indicating that subjects may not have actually used the methods. For example, the heuristics-driven method provided a description of how to identify all verb-object pairs (see Section 3) contained in a statement, as well as all 12 heuristics. Neither the standards-driven nor resource-driven methods included instructions or reference material as lengthy as the heuristics-driven method. Logically, subjects using the heuristics-driven method should have spent *more* time than subjects in the standards- and resource-driven methods when identifying MLEs for a given statement. The subjects using the heuristics-driven should have referred to the 12 heuristics for each action they identified, leading to higher response times.

In addition to providing more training on how to use the different methods for identifying MLEs, we could assess subject's understanding of the method before allowing the subject to complete any of the tasks. If the subject does not understand how to consistently apply the method to complete a given task, then the subject could receive additional training. *For any method for identifying MLEs, if a subject does not understand how to apply the method, he or she cannot be expected to consistently apply the method correctly, which could affect the results of the experiment.*

10.4 Block by Subject Pre-period Performance

We underestimated the variability of subject performance during the pre-period task. By random chance, we observed that subjects had high baseline performance in certain groups. For example, response times for TraditionalRequirements-Complex subjects were higher than for other artifactType*readability groups. In addition, UserManual-Simple subjects had a lower MLE identification correctness during the pre-period than other groups, so the subject performance had more room for improvement during the post-period task than for other treatment groups. *By blocking by subject baseline performance, we could have mitigated the random chance of observing significant interactions due to unusually high or low pre-period task performance.*

11 Threats to Validity

As with all research, the results reported in this paper relate to the context in which the experiment was conducted and may not generalize beyond this context. We have considered the following threats to validity.

11.1 Internal Validity

Selection Subjects were randomly assigned to the 18 different artifact type × readability × method groups. Unbalanced groups in terms of subject expertise could result. However, based on the mean, median, and standard deviations of all metrics for academic and professional experience, and English comprehension of subjects for all combinations of the three factors, groups were evenly balanced in terms of expertise.

Instrumentation When measuring time spent on the task, we automatically recorded time spent in identifying MLEs for each input statement to have a more accurate measure, compared to self-reporting by participants.

11.2 External Validity

Representativeness of Sample Population Falcao et al. describe a systematic mapping study of software engineering experiments that use human subjects (Falcao et al. 2015). The researchers observed that undergraduate students were used in 44.6 % of the experiments, graduate students were used in 33.6 % of the experiments, a mix of students and professionals were used in 11.1 % of the experiments, and only professionals were used in 10.7 % of the experiments. Subjects in our study were enrolled in a graduate course on software security, and the study was conducted at the end of the course. In this sense, the subjects can be considered representative of entry-level, non-expert security practitioners.

Salman et al. (Salman et al. 2015) performed an empirical study to test if student performance differs from performance of professionals on software engineering tasks. The researchers concluded that professionals write higher quality code than students when applying development approaches familiar to both the professionals and students; however, professionals and students performed similarly when applying a development approach in which both professionals and students are inexperienced. Furthermore, Host et al. (Host et al. 2000) found only minor differences between students and professionals in carrying out various software engineering tasks requiring "general understanding of dependencies and relationships". In our study, subjects were asked to apply three new methods with which they had no prior experience. The heuristics-driven technique is intended to be generalizable to all software engineers, regardless of programming experience, security expertise, or domain. No parts of the standards-driven, resource-driven, or heuristics-driven methods for identifying MLEs require programming, security expertise, or domain knowledge. For example, for the heuristics-driven process, subjects need only a basic understanding of English grammar for identifying verbs and objects, then considering twelve heuristics that describe situations in which certain actions (verbs) should or should not be logged.

Experimental Constraints that Limit Realism Subjects could use a maximum of 3 min per input statement to identify MLEs, which may affect the quantity and completeness of the identified MLEs. Similarly, subjects had to first read and understand how to use a given method for identifying MLEs before they could begin the post-period task. To help mitigate this threat, we did not impose a time limit for reading and understanding how to use a given method. Subjects only began the post-period task when they felt that they had learned the method and chose to continue to the post-period task.

11.3 Construct Validity

Mono-operation Bias We selected the tasks from three different software artifact types and multiple levels of readability within each artifact type. Thus, we mitigate the threat of introducing biases due to familiarity of subjects with a particular type of artifact. However, even though most subjects were unfamiliar with the iTrust use-case based requirements specification, all subjects *were* familiar with the actual iTrust software system web-based interface. The iTrust web-based interface was used throughout several exercises and activities involved in the software security course in which the subjects were enrolled.

11.4 Conclusion Validity

Reliability of Measures We minimized biases during the evaluation of the responses by employing quantitative measures, having multiple independent evaluators, and using an oracle of MLEs created beforehand.

Number of Subjects Falcao et al. describe a systematic mapping study of software engineering experiments that use human subjects (Falcao et al. 2015). The researchers found 109 software engineering experiments in which the number of subjects per experiment ranged from 4 to 270, with a mean of 41, median of 26, and standard deviation of 9.7. In our experiment, we include 103 total subjects. Increasing the number of subjects would only lower the standard errors of our estimates to achieve statistical significance (that the effect is different from 0). The estimated effect sizes for the insignificant effects, which were not reported due to space, were small enough that they would not yield any significance. We conjecture that the small effect sizes were due to the time constraints of the experiment. We, therefore, recommend that future experiments adopt a protocol that maximizes the opportunity for the subjects to absorb and appropriately use the assigned methodology (see Section 10).

12 Conclusion

In this study, we conducted a controlled experiment to evaluate the effectiveness of three different methods for identifying MLEs, including a standards-driven, resource-driven, and our previously-proposed heuristics-driven method based on empirically-derived heuristics (King et al. 2015). We evaluate the three methods using statements from three different types of software artifacts and two levels of readability scores. We compare the methods in terms of statement classification correctness, statement MLE identification correctness, and statement response time in identifying MLEs for a given statement. Based on the results of our statistical analysis, we found that no significant difference exists among the three methods in terms of statement classification correctness, statement MLE identification correctness, or statement response time. For future replications of the experiment, we suggest providing both verbal and written instructions, removing all time constraints, and providing more training on how to use each of the methods for identifying MLEs.

Acknowledgments This work is funded by the United States National Security Agency (NSA) Science of Security Lablet. Any opinions expressed in this report are those of the author(s) and do not necessarily reflect the views of the NSA. We thank the Realsearch research group for providing helpful feedback on this work. This study was approved by the North Carolina State University Institutional Review Board (#5354).

References

Carletta J (1996) Assessing agreement on classification tasks: the kappa statistic. Computat Linguist 22(2):249–254, http://dl.acm.org/citation.cfm?id=230386.230390

Chuvakin A, Peterson G (2010) How to do application logging right. IEEE Secur Priv 8(4):82–85. doi:10.1109/MSP.2010.127

Falcao L, Ferreira W, Borges A, Nepomuceno V, Soares S, Baldassare M (2015) An analysis of software engineering experiments using human subjects. In ACM/IEEE International Symposium on Empiriral Software Engineering and Measurement. Beijing, China

Fu Q, Zhu J, Hu W, Lou J-G, Ding R, Lin Q, Zhang D, Xie T (2014) Where do developers log? An empirical study on logging practices in industry. In Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014, 24–33. New York, New York, USA: ACM Press. doi:10.1145/2591062.2591175. http://dl.acm.org/citation.cfm?id=2591062.2591175

Health Insurance Reform: Security Standards (2013) United States Department of Health & Human Services. http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/securityrulepdf.pdf

Host M, Regnell B, Wohlin C (2000) Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. Empir Softw Eng 5(3):201–214

Kincaid JP et al (1975) Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for navy enlisted personnel. (January 31). http://eric.ed.gov/?id=ED108134

King J, Williams L (2013) Cataloging and comparing logging mechanism specifications for electronic health record systems. (August 12): 4. http://dl.acm.org/citation.cfm?id=2696523.2696527

King J, Smith B, Williams L (2012) Modifying without a trace: general audit guidelines are inadequate for open-source electronic health record audit mechanisms. In 305–314. IHI '12. ACM. doi:10.1145/2110363.2110399. http://doi.acm.org.prox.lib.ncsu.edu/10.1145/2110363.2110399

King J, Smith B, Williams L (2012b) Audit mechanisms in electronic health record systems: protected health information may remain vulnerable to undetected misuse. Int J Comput Models Algorithm Med 3(2):23–42. doi:10.4018/jcmam.2012040102

King J, Pandita R, Williams L (2015) Enabling forensics by proposing heuristics to identify mandatory log events. In Proceedings of the 2015 Symposium and Bootcamp on the Science of Security - HotSoS '15, 1–11. New York, New York, USA: ACM Press. doi:10.1145/2746194.2746200. http://dl.acm.org/citation.cfm?id=2746194.2746200

McKenney M (2014) Additional consideration of prior conduct and performance issues is needed when hiring former employees. United States Treasury Inspector General for Tax Administration. http://www.treasury.gov/tigta/auditreports/2015reports/201510006fr.pdf

McLaughlin GH (1969) SMOG grading: a new readability formula. J Read 12(8):639–646

Meneely A, Smith B, Williams L (2012) iTrust electronic health care system: a case study. In: Software and systems traceability. Cleland-Huang J, Gotel O, Zisman A (eds.) Springer

Payment Card Industry Data Security Standards (2010) Paymenet card industry security standards council. https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf

Family Educational Rights and Privacy (1974) United States: United States Government Publishing Office. http://www.ecfr.gov/cgi-bin/text-idx?rgn=div5&node=34:1.1.1.1.33

Read R (2015) University of oregon unlawfully releases 22,000 pages with confidential faculty, staff and student records. The Oregonian/OregonLive. http://www.oregonlive.com/education/index.ssf/2015/01/university_of_oregon_illegally.html

Riaz M, King J, Slankas J, Williams L (2014) Hidden in plain sight: automatically identifying security requirements from natural language artifacts.In: *2014 I.E. 22nd International Requirements Engineering Conference (RE)*, 183–192. IEEE. doi:10.1109/RE.2014.6912260. http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6912260

Riaz M, Slankas J, King J, Williams L (2014) Using templates to elicit implied security requirements from functional requirements - a controlled experiment. In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14, 1–10. New York, New York, USA: ACM Press. doi:10.1145/2652524.2652532. http://dl.acm.org/citation.cfm?id=2652524.2652532

Roper E (2013) Driver's license snooping gets costly for taxpayers. StarTribune. http://www.startribune.com/local/west/220066801.html

Salman I, Misirli A, Juristo N (2015) Are students representatives of professionals in software engineering experiments?" In Proceedings of the 37th International Conference on Software Engineering 666–676

Shull F, Singer J, Dag IK, Sjøberg (2008) Guide to advanced empirical software engineering. Springer, London. doi:10.1007/978-1-84800-044-5

Smith EA, Senter RJ (1967) Automated Readability Index. AMRL-TR. Aerospace Medical Research Laboratories (6570th) (May): 1–14. http://www.ncbi.nlm.nih.gov/pubmed/5302480

Solingen V, Basili V, Caldiera G, Romback HD (2002) Goal Question Metric (GQM) approach. Encyclopedia of Software Engineering

Standard Specification for Audit and Disclosure Logs for Use in Health Information Systems (2013) ASTM International. http://www.astm.org/Standards/E2147.htm

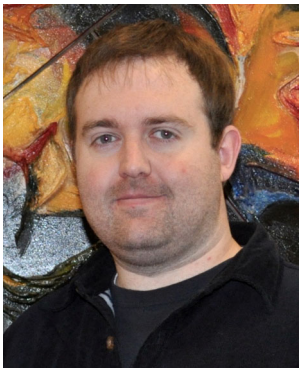
Survey of Patient Privacy Breaches (2011) Veriphyr Inc. http://www.veriphyr.com/landing/HIPAA_violation_survey/

Vance A, Lowry PB, Eggett D (2013) Using accountability to reduce access policy violations in information systems. J Manag Inf Syst 29(4):263–290. doi:10.2753/MIS0742-1222290410

Yskout K, De Win B, Joosen W (2008) Transforming security audit requirements into a software architecture. In: CEUR Workshop Proceedings. Sun SITE Central Europe CEUR-WS. Vol. 413

Yuan D, Park S, Zhou Y (2012) Characterizing logging practices in open-source software. In 2012 34th International Conference on Software Engineering (ICSE), 102–112. IEEE. doi:10.1109/ICSE.2012.6227202. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6227202

Yuan D, Zheng J, Park S, Zhou Y, Savage S (2012b) Improving software diagnosability via log enhancement. ACM Trans Comput Syst 30(1):1–28. doi:10.1145/2110356.2110360



Jason King is a Teaching Assistant Professor in the Department of Computer Science at North Carolina State University. Jason's research focuses on software security and user activity logging mechanisms for increasing accountability and forensicability following security or privacy breaches. He received his PhD in Computer Science from North Carolina State University.

Jon Stallings is an Assistant Professor in the Department of Statistics at North Carolina State University. His main research area is the design and analysis of experiments. His most recent work has focused on developing criteria to compare designs that allow researchers to tailor design selection to their specific research goals. Dr. Stallings also has extensive experience in collaborating with researchers with other fields including entomology, biomechanical engineering, and computer science. He received his PhD in Statistics from Virginia Polytechnic Institute and State University.

Maria Riaz has recently completed her PhD in Computer Science from North Carolina State University. Her research interests include software engineering, software patterns, security requirements engineering, empirical methods in software engineering and distributed computing. She has taught undergraduate courses, supervised senior design projects and served as the advisor for Women Engineering Society for over four years as a lecturer prior to her PhD.

Laurie Williams is a professor and the associate department head in North Carolina State University's Department of Computer Science. Laurie is a co-director of the NCSU Science of Security Lablet. Laurie's research focuses on software security particularly in relation to healthcare IT; agile software development practices and processes; and software reliability, software testing and analysis. She received her PhD in Computer Science from the University of Utah.