A distributed requirements management framework for legal compliance and accountability

Travis D. Breaux¹,*, Annie I. Antón¹, Eugene H. Spafford²

¹North Carolina State University, 890 Oval Drive, Raleigh, NC 27695, USA
²Purdue University, CERIAS, 656 Oval Drive, West Lafayette, IN 47907, USA

Abstract

Increasingly, new regulations are governing organizations and their information systems. Individuals responsible for ensuring legal compliance and accountability currently lack sufficient guidance and support to manage their legal obligations within relevant information systems. While software controls provide assurances that business processes adhere to specific requirements, such as those derived from government regulations, there is little support to manage these requirements and their relationships to various policies and regulations. We propose a requirements management framework that enables executives, business managers, software developers and auditors to distribute legal obligations across business units and/or personnel with different roles and technical capabilities. This framework improves accountability by integrating traceability throughout the policy and requirements lifecycle. We illustrate the framework within the context of a concrete healthcare scenario in which obligations incurred from the Health Insurance Portability and Accountability Act (HIPAA) are delegated and refined into software requirements. Additionally, we show how auditing mechanisms can be integrated into the framework and how auditors can certify that specific chains of delegation and refinement decisions comply with government regulations.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

National and international standards, regulations and laws impose restrictions on business practices to achieve societal goals, such as improving corporate accountability in financial markets or ensuring the privacy of medical records in the healthcare industry. Mature standards and regulations describe specific personnel responsibilities that cut across several business units and require comprehensive documentation to demonstrate how personnel decisions implement standards and regulations. Furthermore, certification boards and government auditors impose penalties on organizations to motivate corrective action in the event of non-compliance. While organizations are often held accountable, recent legislation such as Sarbanes–Oxley¹ (SOX) and the Health Insurance Portability and Accountability Act² (HIPAA) in the U.S. shifts liability towards personnel, imposing fines and prison sentences on individuals for their actions that contribute to non-compliance.

Compliance with laws refers to “the ability to maintain a defensible position in a court of law” (Breaux et al., 2006a). To demonstrate compliance, organizations must exercise due diligence, which describes “reasonable efforts that persons make to satisfy legal requirements or discharge their legal obligations”...
Government regulations usually require a set of artifacts that demonstrate and account for personnel actions taken to comply with the law – the matter of accountability. Because business practices often include a significant human factor (e.g., people implementing policies in a potentially ad-hoc fashion), compliance with standards and regulations is complicated by pressures on human performance (e.g., increasing profits, decreasing costs). In large organizations, software systems that support these processes can provide increased compliance assurance by supporting software controls that restrict what actions personnel can perform with oversight under the law. In effect, software provides a means to enable business practices while limiting the improper use of resources that would otherwise violate the law.

Organizations are in need of mechanisms to help assure that operational practices comply with standards and regulations. According to Ernst and Young (2007) survey of over 1100 international organizations, the top two drivers of information security practice are compliance with regulations and data privacy protection. Moreover, companies are often required by regulatory rules, such as HIPAA (US OCR, 2003a,b), to implement policies and procedures that comply with the law.

To meet the needs of operational controls, we propose a distributed requirements management framework. Our framework provides a transparent and accountable method of ensuring that obligations in standards and regulations are implemented by functional software requirements and software controls. In our framework, personnel satisfy obligations by refining them into functional requirements or by creating new obligations that are delegated to others. Through delegation and refinement, personnel will contextualize their obligations incurred from standards and regulations using their own business knowledge and goals. Recording the personnel decisions to delegate and refine obligations improves accountability, because each decision can be evaluated and compared against best practices. Auditors can certify these decision chains to demonstrate that, at least at a specific point in time, organizations complied with acceptable interpretations of the law. Furthermore, as policies change, organizations can re-evaluate their decisions to delegate and refine their legal obligations in a framework that dynamically dispatches these changes to personnel responsible for accommodating these changes. While this work has not yet been validated in practice, it has been mathematically validated using Alloy (Jackson, 2002) and we believe the framework is relevant, effective, and feasible.

In our previous work, we analyzed privacy policies in finance (Antón et al., 2004) and healthcare (Antón et al., 2007; Breaux and Antón, 2005), organizational security policies (Breaux et al., 2006a) and U.S. federal regulations (Breaux and Antón, 2008; Breaux et al., 2006b, 2008) to identify policy elements required to align systems with policies and regulations. In each of these studies, we identified a need to manage obligations in a single, distributed framework. Our proposed framework builds upon this need by managing obligations through delegation and refinement with special focus on the needs of auditors.

This paper is organized as follows: in Section 2, we consider a simple scenario to motivate our framework; in Section 3, we present the framework formalism and definitions; in Section 4, we instantiate our framework by elaborating on an application in the healthcare domain; in Section 5 we discuss related work; in Section 6 we discuss requirements for a tool supporting our framework with our conclusion in Section 7.

2. Requirements scenario

Consider a scenario in which a Chief Security Officer (CSO) has been assigned the high-level security goal (an obligation and non-functional requirement) $\text{OB}_n$ “to ensure that corporate information is secure.” The CSO implements $\text{OB}_n$ by assigning several new non-functional requirements including obligation $\text{OB}_b$ “ensure computer-based communications are confidential” to his information technology (IT) security manager in charge of network security. The IT security manager responds by identifying all modes of “computer-based communications” relevant to satisfying her new obligation. As a result, the manager identifies internal web, instant-messaging and e-mail servers among others that use TCP/IP network connections to share information among internal systems. The manager, with both authority over who administers these servers and knowledge of available security mechanisms in these systems, implements her obligation by assigning new functional requirements including the functional requirement $\text{SR}_a$ “ensure web servers use Secure Sockets Layer (SSL) for internal connections” and $\text{SR}_b$ “ensure mail users use Transport Layer Security (TLS) for internal connections” to relevant system administrators across different departments. These functional requirements, called refinements, constitute her choice in selecting methods and techniques she believes are relevant to comply with $\text{OB}_b$. The inability of any actor to foresee all possible and relevant refinements may lead to potential gaps in compliance. A system administrator responsible for administering a mail server running Linux receives $\text{SR}_b$ and implements the requirement with a series of configuration directives that he applies to the system: $\text{SR}_1$ = “install latest OpenSSL libraries,” $\text{SR}_2$ = “compile and configure Sendmail with TLS support”, $\text{SR}_3$ = “generate X.509 certificates for Sendmail,” etc.

At each level in the delegation hierarchy, a manager knows what goal his staff member must achieve but the manager may not have the technical knowledge to know how his staff will achieve this goal. Each obligation is owned by someone who is ultimately accountable for that obligation and the decisions to refine an obligation are also recorded. Tracing permissions and obligations through ownership, delegation and refinement allows managers and auditors to quickly and effectively identify who and why vulnerabilities are addressed to reduce risk of non-compliance.

3. Management framework

Our proposed distributed requirements management framework provides traceability from the regulations that govern organizations to the decisions of actors who assign obligations to other actors and finally to the software requirements assigned to systems that refine those personnel obligations. Fig. 1 shows the associations maintained in our framework among actors, systems and obligations in the Unified

Please cite this article in press as: Travis D. Breaux et al., A distributed requirements management framework for legal compliance and accountability, Comput. Secur. (2008), doi:10.1016/j.cose.2008.08.001
Modeling Language (UML), an Object Management Group standard that is frequently used to represent software architecture in practice. In the UML notation, boxes represent classes, arrows lead from general classes to specialized subclasses, and diamonds connect from whole classes to their constituent parts. Cardinality constraints appear at the ends of connecting lines: “*” means zero or more, “1” means exactly one. In assignment, also called delegation, each assigned obligation has exactly one owner (an actor) who is ultimately responsible for satisfying the obligation. Ownership can be transferred, but not shared. The owner may assign the obligation to other actors and the owner may further assign those actors to re-delegate the obligation to others.

Because the management framework seeks to mediate between personnel and information systems, we distinguish between two types of obligations: responsibilities that require a person to perform some action and requirements that require a system to have some property or perform some function. For all systems, there is exactly one oversight responsibility that requires exactly one owner to be ultimately responsible for implementing the requirements assigned to those systems. In some situations this actor may be permitted to re-delegate her oversight responsibility for a specific system to other actors. We discuss the important issue of authorizing delegation and refinement decisions in Section 5 on related work.

We now define key terms to elucidate the primary elements of our management model.

### 3.1 Management model

Whereas Fig. 1 presents the framework associations that constitute the model as a software architecture in the UML, this section describes the model using axiomatic set theory. Let the set $A$ consist of actors, the set $S$ consist of systems and the set $O$ consist of obligations.

**Definition 1.** The assignment set $AS \subseteq (A \times S) \times O$ is a many-to-many relation mapping actors and systems to their assigned obligations. Each actor or system may have multiple obligations and each obligation may be assigned to multiple actors or systems. Each act of delegation and refinement yields a new assignment.

**Definition 2.** The delegation set $DS \subseteq A \times AS$ is a many-to-many relation mapping actors (the delegator) to the obligations that they assign to other actors and systems (the delegatee). We assume the delegator $a \in A$ is assigned the obligation $o$, prior to delegation, $\exists <a, o> \in AS$, and, to be consistent, we require that all delegations yield a valid assignment for the delegatee $a \in A$, such that $\forall <a, o, a> \in DS, \exists <a, o> \in AS$. We also assume a permission framework is in-place to ensure that each delegator is authorized to delegate obligations to the chosen delegatee.

**Definition 3.** The ownership set $N \subseteq A \times (O \times S)$ is a one-to-many relation mapping actors to the obligations and systems they own; every obligation and system has exactly one owner. Each owner is solely responsible for monitoring the accountability of the obligations they own. For each system $s \in S$ with system owner $a \in A$ expressed by the ownership pair $<a, s> \in N$, there is one oversight responsibility $o$ and corresponding assignment $<a, o> \in AS$ that requires the system owner to satisfy all the assigned system requirements $r$ in $<s, r> \in AS$.

**Definition 4.** The refinement set $RS \subseteq AS \times O$ is a many-to-one relation mapping actors or systems and their assigned obligations to refinements (other obligations). We assume these refinements are created by the actor who is assigned the obligation or by the system owner. To be consistent, we require that all refinements $o_i$ from obligations $o$, yield a valid assignment for the same actor or system $a$, such that $\forall <a, o_i, o> \in RS, \exists <a, o> \in AS$. Because a single obligation can be delegated to multiple actors, these actors may then refine this obligation into different specialized obligations in ways that satisfy the context of their daily operations. For an actor or system $a \in (A \cup S)$ and an obligation $o \in O$, the set $\{r|<a, o, r> \in RS\}$ is called a refinement strategy. The refinement strategy describes the subset of possible alternative refinements that were chosen by the actor or system owner to refine a specific obligation.

**Definition 5.** The decision sequence $<d_1, d_2, \ldots, d_m> \subseteq (DS \cup RS)$ is derived by tracing an obligation through delegation and refinement decisions. A valid decision sequence is comprised of a series of delegation and refinement sequences: a delegation sequence $<a, a_2, o>, <a_2, a_3, o>, \ldots, <a_m, a_{m+1}, o> \subseteq DS$ of length $m$ traces the obligation $o$ from the delegator $a_1$ to the delegatee $a_{m+1}$, and so on for $1 \leq i \leq m$; and a refinement sequence $<a, o_1, o_2>, <a, o_2, o_3>, \ldots, <a, o_{n-1}, o_n> \subseteq RS$ of length $n$ traces the obligation $o_i$ assigned to the actor $a$ to the refinement $o_{i+1}$, and so on for $1 \leq j \leq n$. A valid decision...
sequence begins with either a delegation or refinement decision $d_1 \in (DS \cup RS)$ and alternates between zero or more delegation and refinement sequences. To clarify how these sequences are connected, for subscripts $u, v, x, y$: a refinement sequence follows a delegation sequence by $\{ \ldots, <a_u, a_o, o>, <a_v, a_o, o>, \ldots \}$ where a delegator $a_u$ delegates an obligation $o$, to a delegatee $a_o$ who refines obligation $o$ into obligation $o_{\alpha}$; and a delegation sequence follows a refinement sequence by $\{ \ldots, <a_u, o_y>, <a_v, o_y>, \ldots \}$ where a delegator $a_u$ refines an obligation $o_y$ into an obligation $o$, and delegates $o_{\alpha}$ to the actor $a_o$. We presently assume that no cycles exist in all sequences of delegations and refinements derivable from $(DS \cup RS)$; in practice, cycles can be identified and avoided. The refinement and delegation of obligations is complete if it is accountable, which we now discuss.

3.2. Accountability

Definition 6. An obligation is accountable if a mechanism exists to verify that the obligation has been satisfied (Breaux et al., 2006a). This mechanism may either be: (1) an oracle (e.g., executable program, hardware device) that returns true if and only if the obligation is achieved or maintained; or (2) the evaluation of a logical expression comprised of a conjunction of predicates in which each predicate denotes the satisfaction of obligations in a refinement strategy that consists of accountable obligations.

Consider, for example, the accountability of system requirements. Because functional requirements are testable by definition, some data or program exists called a test case that may be used to verify whether a system satisfies those requirements – hence, functional requirements are always accountable. Non-functional requirements are not testable in the same fashion, but they are accountable if they are refined into functional requirements. Therefore, testing non-functional requirements is tantamount to testing their refinements, assuming each refinement is either itself a functional requirement or another non-functional requirement that is refined into one or more other accountable requirements. Because delegation can transfer obligations from one actor to another, testing accountability in this distributed framework relies upon valid decision sequences.

Personnel responsibilities may also be accountable within this framework, if they are supported by software systems that retain sufficient information to evaluate those responsibilities. For example, the obligation “to only use a password that contains at least eight characters” is accountable by executing a program to check the length of a user’s personal password when it is set. However, the responsibility “to logout from a system when the system is no longer in use” is not accountable, as it is difficult to define the behavior of “in use” for all users, systems and applications.

Definition 7. The verification set $VS \subseteq \text{Boolean} \times AS$ is a one-to-one mapping of Boolean predicates to assignments such that each predicate is true if and only if the assigned obligation is satisfied by the actor or system in the assignment. For a verification $<u, o, \alpha> \in VS$, the predicate $\alpha$ evaluates to either: (1) true if an oracle or test case decides the obligation is satisfied, or false otherwise; or (2) the logical conjunction of predicates assigned to some number of refinements: for the verification $<u, o, \alpha> \in VS$, let $V = V_0 \land V_1 \land \ldots \land V_n$ such that $<u, o, \alpha> \in RS$. The expression must contain exactly those predicates for refinements $0$ that are necessary and sufficient to satisfy the obligation $o$.

There will be situations in which only a human being can verify whether or not an obligation has been satisfied. In the HIPAA for example, there are situations in which medical information can only be shared with third parties in the event of a medical emergency. Because medical emergencies can only be determined by appropriate individuals, an information system can at best receive this determination from a human but not verify the emergency itself. The framework can easily be extended to identify these oracles that receive these determinations. Moreover, auditing mechanisms can be put in-place to maintain logs of these oracles to identify misuse and for forensic analysis after an abuse of the system (Buchholz and Spafford, 2004; Damianou et al., 2001).

4. Applying the framework

We apply our framework from Section 3 to an example in which obligations (OB) from the HIPAA Privacy (US OCR, 2003a) and Security (US OCR, 2003b) Rules are delegated from upper management to their staff and later refined into software requirements (SR). At each delegation stage, an employee with specialized responsibility and technical expertise interprets his or her assigned obligations and refines and/or re-delegates these obligations, as needed. We illustrate this application by narrating the sequence of delegation and refinement decisions. In each state, we list the obligations followed by the expressions in our model that record these decisions. In addition to showing how our prototype framework would be applied to a real set of requirements taken from HIPAA, this example serves to show some of the complexity — and subtleties — present in legal requirements.

In the following example, we use the delegation mechanism to map obligations from policies and regulations to actors who must satisfy these obligations. These assignments coincide with the delegatee becoming the owner of those obligations. For example, the Chief Security Officer (CSO) for a healthcare provider (a covered entity) is assigned the following obligation from the HIPAA Security Rule (HSR) §164.308(a)(2):

- **OB**$_1$: identify the security official who is responsible for the development and implementation of the policies and procedures required by the HIPAA section 164 subpart C for the covered entity.

  \[
  <\text{HSR}, \text{CSO}, \text{OB}_{1},> \in DS
  \]

  \[
  <\text{CSO}, \text{OB}_{1}> \in N
  \]

To satisfy obligation **OB**$_1$, the CSO identifies the appropriate security official (SO). Consequently, this role incurs obligations from the Security Rule §164.302–§164.318 to the SO, including:

- **OB**$_2$: from §164.312(a)(1): allow access only to those persons or software programs that have been granted access rights as specified in §164.308(a)(4);
OB2: from §164.308(a)(4): authorize access to electronic protected health information (PHI) that are consistent with the applicable requirements of subpart E of this part (e.g., the Privacy Rule); and

OB2: §164.308(a)(4)(ii)(B): grant access to electronic PHI, for example, through access to a workstation, transaction, program, process, or other mechanism.

[{<SR1>, SO, OB2}, <SR1>, SO, OB2, <SR1>, SO, OB2] \in DS

Notably, the assignment of obligation OB2 to the SO incurs authority authorized by law to receive reports of child abuse and

OB2: §164.312(e)(1): guard against unauthorized access to electronic PHI that is transmitted over an electronic communications network; and

OB2: §164.312(e)(1)(ii): encrypt electronic protected health information whenever deemed appropriate.

[{<SR1>, SO, OB2}, <SR1>, SO, OB2] \in DS

[{<SO, OB2}, <SO, OB2}] \in N

Furthermore, for those authorizations in the Privacy Rule that participate in a transaction and utilize an "electronic communications network," the SO must also implement technical measures to:

OB2: in §164.312(e)(1): guard against unauthorized access to electronic PHI that is transmitted over an electronic communications network; and

OB2: from §164.512(b)(1)(ii): disclose PHI to a government authority authorized by law to receive reports of child abuse or neglect.

OB2: from §164.512(c)(1)(ii): except for reports of child abuse or neglect permitted by paragraph (b)(1)(ii), disclose PHI about an individual whom the covered entity reasonably believes to be a victim of abuse, neglect, or domestic violence to a government authority authorized by law to receive reports of such abuse, neglect, or domestic violence ...

The ISA surveys existing systems within the covered entity and assigns the four requirements SR1–SR4 to relevant systems. If existing systems are unable to satisfy any requirements, those requirements are assigned to a software engineer who will design, develop, test and deliver a new system or configuration to meet these requirements.

4.1. Compliance and accountability

Heterogeneity between business practices in different organizations makes it difficult to develop a single, de-facto implementation of standards and regulations to achieve compliance. Consequently, auditors and external reviewers must certify that a set of business practices comply with a set of standards or regulations at a specific point in time: a process called certification. In addition, auditors must acquire real world evidence demonstrating that business practices continue to comply, either through random or continuous sampling of appropriate data: this acquisition is called an audit, and is performed as a form of compliance monitoring. In the event that business decisions are made to comply with laws that are unclear, what constitutes compliance may require legal opinion.

Using our framework, organizations can exhibit decision sequences that trace regulations to functional requirements. Each requirement is verified using one or more test cases to verify software systems. Auditors and external reviewers may certify that these sequences comply with law using a digital or cryptographic signature. The function sign: (K × M → S maps secret keys K and messages M to a unique cryptographic signature in S. The signature can be used to verify that a message, in this case the decision sequence, has not changed and, furthermore, to digitally identify the auditor or legal expert who certified the sequence. The digital signature
does not verify the credibility or authority of the auditor to certify a decision sequence. In follow-up certification reviews or as expert legal opinions change, decision sequences are re-certified using the history of verification predicates obtained during the policy and runtime requirements lifecycle.

Returning to the application of our framework in Section 4, the auditor identifies the following decision sequence \( D = \{d_1, d_2, d_3, d_4\} \):

\[
\begin{align*}
\{d_1 &= \{<HSR, \text{SO}, \text{OB}_2>, d_2 = \{<\text{SO}, \text{ISA}, \text{OB}_2>\} \in DS \\
\{d_3 &= \{<\text{ISA, OB}_7, \text{SR}_1>, d_4 = \{<\text{ISA, SR}_a, \text{SR}_1>\} \in RS
\end{align*}
\]

The ISA exhibits the verification subset \( V \subseteq VS \) consisting of verifications for all systems \( s \) in \( <s, \text{SR}_1>, <s, \text{SR}_2> \in AS \). The auditors, deciding that the sequence \( D \) and verification set \( V \) are all necessary to comply with the regulations \( \{\text{OB}_a, \text{OB}_b\} \) at time \( t \), use their secret key \( k \) to provide the certificate \( C = \{\text{sign}(k, D, V + t), D, V, t\} \). In subsequent reviews at time \( t' \geq t \), an auditor re-certifies these systems by verifying \( D \) and \( V \) against the previous certificate and, if the auditor accepts the result, the auditor will issue a new certificate updated for time \( t' \). A series of these certificates for a single regulation at different time intervals is the compliance history for that regulation.

In the event of a violation of law or regulation, our framework provides a starting point for investigation. The violated obligation or requirement is identified, and then traced through to implementation and operation. Involved parties can also be identified from among all those associated with system development and operation. The result may be a faster, more agile response to addressing violations.

5. Related work

In requirements engineering, traceability is a measure of quality that reduces the risk of not propagating relevant changes to artifacts throughout the development lifecycle. Maintaining traceability information can be time consuming (Dömges and Pohl, 1998; Ramesh, 1998); thus, some may view traceability as too costly. However, traceability reduces the risk of inconsistencies and ensures compliance of a resulting system with the software requirements specification (Antón et al., 2001). Within the context of regulatory compliance, traceability increases in significance and criticality because requirements are driven by multiple policies distributed throughout an organization. Introducing additional traceability within the context of compliance and accountability can affect the cost of these systems. However, the gains and benefits well outweigh the costs (Ramesh, 1998).

Research in system security has shifted focus to include broader issues in organizational management. We see evidence for this shift in both access control frameworks limited to permissions (Bandmann et al., 2002; Barka and Sandhu, 2000; Oh and Sandhu, 2002; Park et al., 2004) and more progressive approaches that also include obligations and delegation (Dulay et al., 2001; Minsky and Ungureanu, 2000; Moffett and Sloman, 1991; Park and Sandhu, 2004). Permissions describe what people and systems are required to do given specific restrictions or constraints. In delegation, a person delegates her authority or responsibility to another person; the latter person acts on behalf of the former.

The proper authority to delegate obligations is important to ensure the standards and regulations can be properly refined into functional requirements. Barka and Sandhu (2000) consider modes of delegation including permanent and temporary delegation. Revoking delegation authority would first require evaluating the impact on obligations that were refined and/or re-delegated. Bandmann et al. (2002) propose constrained delegation as means to moderate delegated authority in a distributed system. Constrained delegation provides a means to control whom a delegator can assign new obligations; this is important to prevent a delegator from obligating actors or systems that are beyond the delegator’s scope of authority.

With regard to permissions, Oh and Sandhu use business units and organizational hierarchy to administer roles and permissions for users (Oh and Sandhu, 2002). Park et al. (2004) attempt to align organizational structure with system structure to improve role-based access control implementations (Park et al., 2004). These approaches highlight the organizational need to conceptually align existing authorization frameworks with organizational structure.

Moffett and Sloman (1991) introduced the concept of policies and system objects, which they later realized in the Ponder language (Damianou et al., 2001) to express authorizations and obligations for managing large networks. A deployment model for distributed network management was proposed using Ponder (Dulay et al., 2001). Minsky and Ungureanu (2000) introduced Law-governed Interactions (LGI) in which actors suffer penalties if they violate their obligations. In LGI, actors subscribe to a shared controller that audits their behavior to detect non-compliance. Park and Sandhu (2004) propose UCONABC to manage authorizations and obligations using conditions for digital rights management. Each of these approaches shares common elements relevant to regulatory compliance, including the ability to express permissions, obligations and delegations and the means to audit compliance through obligations. Moffett identified the need for requirements in policy models (Moffett, 1999). For regulatory compliance, the refinement decisions of actors are also needed to completely trace from regulations to the requirements of systems that satisfy those regulations.

In requirements engineering, related work has focused on goal refinement (Antón, 1997; Antón et al., 1994; Bandara et al., 2004; Darimont et al., 2005; Dardenne et al., 1993) and delegation (Breaux et al., 2006b; Giorgini et al., 2005). Goals describe desired states or actions performed by actors without specific consideration for normative positions (e.g., permissions, recommendations and obligations.) Similar to obligations, goals are decomposed into sub-goals intended to achieve the original goal (Antón et al., 1994). Darimont et al. (2005) described the GRAIL tool that implements the KAOS framework (Dardenne et al., 1993) for modeling goal refinement hierarchies using logical and/or relations and temporal logic. Regulatory compliance complicates the collaborative environment in which obligations are refined by personnel across an organization. Whereas the GRAIL tool goes far to
address the rich semantics of goal refinement, it does not address the broader traceability issue where individual personnel decide how and when to refine obligations. Antón (1997) shows how non-functional requirements can be implemented through functional requirements; a notion captured in our definition of refinement. Bandara et al. (2004) propose applying goal refinement to policies using Event Calculus for temporal reasoning. Breaux et al. (2006b) show how to acquire and model delegation requirements from the HIPAA Privacy Rule and show how these are used to infer new implied rights and obligations. Mylopoulos et al. describe the Secure Tropos framework that models ownership and delegation and defines obligation as “trust in execution” (Giorgini et al., 2005). Similar to GRAIL, Secure Tropos provides a single-user perspective on goal refinement, whereas our compliance framework incorporates multiple viewpoints through distributed refinement. While goal refinement is not new, tracing refinement and delegation of obligations, together, in a distributed environment that supports audits and external reviews provides new opportunities to explore compliance-related issues.

Rees et al. (2003) describe an information security framework, named PFIREs, which combines policy assessment and review to mitigate risk in organizational security. Although the PFIREs framework does not address policies as system objects, per se, the authors propose improving security by passing messages between personnel to communicate obligations and monitor compliance; these messages are necessary to implement a compliance framework such as ours.

The compliance auditing procedure we describe, using cryptographic signing, is similar to a current use case of the Tripwire tool (Kim and Spafford, 1994) in government and financial settings: if the signature matches a saved signature, the code has not deviated from that previously certified.

Finally, several publications on policy, including the U.S. National Institute of Standards and Technology (NIST) Security Handbook (US NISM, 1995), define policies as comprised of standards, guidelines and procedures. Standards are implementable obligations assigned to personnel and systems whereas guidelines are recommendations that may coincide with best practices for specific contexts. Procedures implement standards, often with a step-by-step description that is sufficient for other actors to reproduce the desired results. In our framework, these notions are complementary and can be easily supported. Guidelines require distinguishing the modality of recommendations (should) from obligations (must, shall) and “a procedure” requires conditionally sequencing a set of refinements. Despite such extensions, it is important to emphasize that standards, guidelines and procedures typically originate with upper management, whereas our framework specifically supports middle and lower managers in their effort to contextualize organization-wide goals as obligations and requirements in a business unit. In addition, the ISO 9001 standard on quality requirements management is complementary with its process validation and certification guidelines (ISO/IEC 9001, 2000) and the U.K. Office of Government Commerce (OGC) Information Technology Infrastructure Library (ITIL) provides complementary guidance on managing change and rectifying faults (UK OGC, 2007), two important facets of a comprehensive compliance framework.

6. Requirements for tool support

Building on our previous work in analyzing financial (Antón et al., 2004) and healthcare policies (Antón et al., 2007; Breaux and Antón, 2005), standards (Breaux et al., 2006a) and regulations in privacy and security (Breaux and Antón, 2008; Breaux et al., 2006b; Spafford and Antón, 2007) and accessibility (Breaux et al., 2008), we propose requirements for a tool to support personnel from Chief Security Officers (CSOs) and business managers to software developers and system administrators. The work of CSOs is to organize and delegate the high-level goals that are (usually) relatively few in number and change infrequently, whereas, system administrators must respond to numerous system requirements that adapt systems to emerging business needs and security vulnerabilities. Furthermore, compliance officers and auditors need to evaluate the delegation and refinement decisions made by those personnel.

We acknowledge the inherent challenges in maintaining a tool of this proportion for large organizations and assert that such a tool must itself be subject to certification using an established software security standard, such as the Common Criteria (ISO/IEC 15408, 2005). For small and medium size organizations, the trade-off between introducing new complexity into existing business practices and demonstrating due diligence must be weighed against the risk of non-compliance. While a small or medium size organization might benefit from conducting business in a single, well-defined domain, such as finance or healthcare, they often lack extensive legal resources that are necessary to exhaustively avoid questionable interpretations of law. Consequently, our framework provides small and medium size businesses the tools to rationalize their business practices in preparation for receiving sound legal guidance.

The following is a minimal set of functional software requirements (FR) for a tool that implements our framework. Following the requirements, we briefly describe how these requirements would interact with users as well as future extensions to the framework that would extend the capabilities of the tool.

1. Investigation/audit:

   FR1: the tool shall allow each user to view the “context” of an obligation. The context includes:
   (1) The delegator who assigned the obligation to another actor;
   (2) The obligations, if any, that the assigned obligation refines (e.g., the overall goal the obligation is intended to achieve);
   (3) The refinements, if any, to the assigned obligation proposed by the delegatee; and
   (4) For a sequence of delegations and refinements, any certifications verified by digital signatures and the identities of signatory actors.

   FR2: the tool shall allow each user to separately view any obligations that are assigned to him by other actors.

   FR3: the tool shall allow each user with an oversight responsibility to separately view the requirements assigned to the concerned systems.
Finally, software products could be distributed with their own requirements plug-ins. System administrators who bring a new product online would receive a list of requirements certified by the product developer, based on actual developer test cases. Whereas organizations that develop their own software assume their own liability for resulting software failures, these third-party certifications would establish third-party liability against explicit product requirements within the distributed management framework. Given a non-compliance event associated with a specific decision chain involving a third-party product, both the organization and third-party can determine the role the product played in the fault: the product may have failed to satisfy its requirements, or the requirements may have been inappropriately aligned with personnel obligations they could not reasonably satisfy.

7. Conclusion

In summary, we present a framework that combines delegation and refinement in a distributed system to capture the decisions that executives, managers, system administrators, and developers make to achieve compliance with standards and regulations. The contributions in this work include a formal definition of decision chains that auditors can certify and review when determining compliance for an organization. Furthermore, we instantiate our framework using an example from the HIPAA Security and Privacy Rules and propose requirements for a tool to implement the framework with discussion of its use.

Our framework provides new structure that combines traceability and digital identity (via cryptographic signatures) with the domains of policy, law, software development, and administration. Combined with our observations and discussions with practitioners, these preliminary results suggest the feasibility and effectiveness of the framework. However, relying solely on a tool carries specific risks. These risks must be augmented by meetings and ongoing discussions within organizations that use the tool (Damian et al., 2006). The framework provides important traceability data on policy and requirements decisions that can be used to initiate and drive these discussions. Moreover, these discussions will likely yield interesting insight into additional interactions between requirements, policies and law that the framework does not currently capture. How can these interactions be described formally and integrated into the framework? Can we distill best practices for requirements engineers and auditors from practical applications of the framework? We envision answering these questions by employing a lightweight, cross-platform tool based on the framework in future case studies.

Acknowledgements

This work was funded, in part, by NSF Cyber Trust grant #0430166: Collaborative Research: Comprehensive Policy-Driven Framework for Online Privacy Protection: Integrating IT, Human, Legal and Economic Perspectives and the Purdue Center for Education and Research in Information Assurance and Security (CERIAS).
REFERENCES


Moffett JD, Requirements and policies. In: Workshop on policies for distributed systems and networks, Bristol, UK; 1999.


Travis D. Breaux received the BA degree in Anthropology from the University of Houston in 1999 and the BS degree in Computer Science from the University of Oregon in 2003. He is currently working toward the PhD in Computer Science at North Carolina State University. His research includes
requirements engineering for software systems that are regulated by policies and laws governing privacy, security and accessibility. He is a recipient of the 2006–2009 IBM PhD Fellowship, 2006 Walker H. Wilkinson Research Ethics Fellowship, and 2005 CISCO Information Assurance Scholarship. He is a member of the IEEE Computer Society, ACM SIGSOFT and the ACM US Public Policy Committee.

Annie I. Antón received the PhD degree in Computer Science from the Georgia Institute of Technology, Atlanta, in 1997. In 1998, she joined the faculty of North Carolina State University where she is currently a Professor of Computer Science, the founder and director of ThePrivacyPlace.org, and a member of the Cyber Defense Laboratory. Her research interests include software requirements engineering, information privacy and security policy, regulatory compliance, software evolution, and process improvement. She is a Co-chair of the Privacy Subcommittee of the ACM US Public Policy Committee. She is a member of the ACM, CRA Board of Directors, IAPP, Sigma Xi, and the US Department of Homeland Security Data Privacy and Integrity Advisory Committee. She is a senior member of the IEEE and a recipient of the US National Science Foundation Faculty Early Career Development (CAREER) Award.

Eugene H. Spafford is a professor of computer science at Purdue University. He is also the Executive Director of the Center for Education and Research in Information Assurance and Security (CERIAS). At Purdue since 1987, Spaf is a Fellow of the IEEE, the ACM and the AAAS. He is a past recipient of the NIST/NSA Computer Systems Security Award, the IEEE Taylor Booth Medal, an honorary CISSP, induction into the ISSA Hall of Fame, and the ACM President’s Award, among other recognitions for his work. His current research is directed towards cyber security and forensic issues, and national security policies.