Normative Requirements in Sociotechnical Systems:

NoReST as a Formal Framework for Verification and Refinement

Özgür Kafalı, Nirav Ajmeri, and Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
{rkafali,najmeri,singh}@ncsu.edu

***Abstract*—We address the problem of verifying a sociotechnical system (STS) specification with respect to its requirements. Previous work on verification of software systems either ignores the social dimension, or assumes that autonomous entities (e.g., agents) can be controlled, thereby limiting their autonomy. *Our goal is to systematically refine an STS specification so as to ensure it satisfies stated requirements.* We propose a formal framework, NoReST, which treats an STS as a two-tier normative system. NoReST supports the verification of functional, security, and privacy requirements via model checking. The technical tier provides regimentation via control mechanisms that describe how agents interact with software components, whereas the social tier provides regulation via norms that characterize the expectations of agents from each other. If a requirement is not satisfied, NoReST suggests refinements based on normative patterns. We describe NoReST, provide its implementation using the NuSMV model checker, and demonstrate how it applies in an emergency health care scenario from HIPAA. We show via a modeler study that a design process based on NoReST's normative patterns is helpful for subjects who are inexperienced in conceptual modeling and norms.**

***Index Terms*—Norms, privacy, model checking, patterns**

I. Introduction

We understand a sociotechnical system (STS) as a social organization, wherein autonomous agents representing stakeholders interact with each other through and about technical components [4], [6], [21]. In our conception, an STS supports interaction across two levels: (i) among the agents regarding the applicable social norms, and (ii) between the agents and the technical components that belong to, and are subject to control by, the underlying technical architecture.

Figure 1 illustrates this conception. At runtime (right), an STS's technical tier comprises software components supporting various actions. Its social tier comprises autonomous agents being regulated by norms as they interact with each other and with the technical elements. The technical mechanisms embody hard requirements and allow or prevent specific actions. Following Chopra et al.'s [4], [6] criticism of traditional approaches that they interfere with agent autonomy, we place norms [21] as central to an STS. Agents being autonomous can violate the norms but each norm specifies who is accountable to whom for what and when.

At design-time (Figure 1 left), we begin from stakeholders with requirements: needed functionality and security and privacy. A requirements engineer elicits these requirements and specifies the technical tier (mechanisms) and social tier (norms) along with any domain assumptions. A well-conceived STS would capture ways in which to recover from violations, e.g., by sanctioning an accountable party for a violation. For target applications, such as health care, incorporating norms is essential to gain the flexibility needed to satisfy stakeholder requirements that would otherwise be left unsupported.

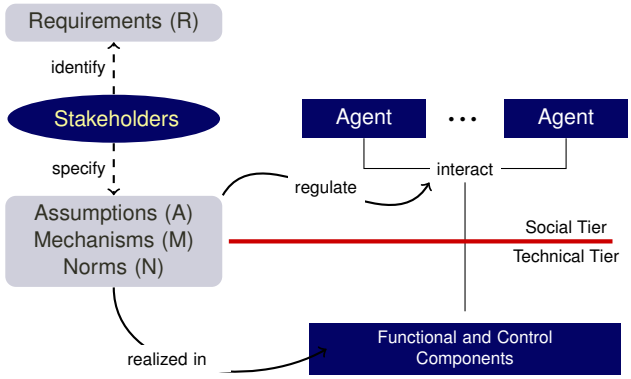


Fig. 1. Conception of an STS.

Running Example: HIPAA Emergency Rule [11], [12] This STS supports regulations regarding the disclosure of patient information in emergency situations and disasters. We adopt distinct typefaces for STAKEHOLDERS and propositions.

Example 1. An emergency physician (EP) can enter the emergency department (ED) by swiping her card [Ex1]. In order to access a patient's electronic health records (EHR), EP should log into the computer in ED using her credentials [Ex2]. She does not need additional consent from the patient to review their EHR. However, she should not disclose the protected health information (PHI) of the patient [Ex3], nor should she access any patient's EHR other than those she is treating [Ex4]. When EP is done reviewing the EHR, she logs off from the computer [Ex5]. In case of a national disaster, requirements about accessing and disclosing the patient's PHI to the patient's family are waived [Ex6].

We now informally apply Figure 1's conception to Example 1. The agents are EP, ED, and PATIENT. The EHR software belonging to ED implements access control mechanisms. For

example, the action of accessing a patient's EHR is enabled by a mechanism if the physician logs in to the software with her credentials. An example norm is a prohibition of EP by ED from disclosing a patient's PHI.

The intuitive benefits of the STS conception are obvious. It captures the social and technical elements together that are traditionally separated (whereby the social tier is given short shrift). This conception brings an important research challenge to fore: What computational representations and techniques can help realize the above vision? We refine this challenge into two major research questions that this paper addresses.

Research question Q_1: Verification in STS. How can we verify that an STS, via its technical and social elements, satisfies the functional, security, and privacy requirements of its stakeholders? Significance: Situations in which agents stand in for autonomous stakeholders are especially crucial to security and privacy, because security and privacy concerns presuppose that multiple autonomous agents are involved. Novelty: Previous research on verification of software systems falls into two bodies. Formal methods [8], [15] for modeling and synthesis of software components do not incorporate autonomous agents into their architectures. Works that incorporate agents [2] artificially limit the agents' autonomy and assume they always comply with the applicable norms.

Research question Q_2: Refining an STS. How can we refine an STS's specification to ensure it satisfies stated requirements? Significance: Creating specifications is at the heart of software engineering; doing so for STSs with first-class status for the social tier can help avoid loss of user requirements via ad hoc translation. Accommodating requirements changes in an STS too is nontrivial. Novelty: Existing approaches on STSs, e.g., Protos [4], provide an abstract design process but no concrete support for producing specifications.

Contributions and organization. We propose NOREST, a formal framework for the verification and refinement of STS specifications. via social norms (Section II), and supports formal verification of requirements via the NuSMV model checker by generating a model based on the correct behavior described by the norms (Section III). In case an STS specification does not satisfy a stated requirement, NOREST supports refinement based on normative patterns that make specifications more or less flexible (Section IV). NOREST is novel because it incorporates the social dimension, which has not been considered by previous work. We develop WICKED, a tool for NOREST, that provides a user interface for the modeling, configuration, verification, and refinement of an STS. We demonstrate how refinement works on an emergency healthcare scenario from HIPAA (Section V). We conduct an empirical study of modelers. Analysis of our study shows that a design process with NOREST's patterns is helpful for subjects with less experience in conceptual modeling and norms (Section VI).

II. FORMAL FRAMEWORK

We now describe the formal constructs that make up the technical and social tiers of an STS. Our universe of discourse comprises three finite sets: (1) $\mathcal{R} = \{\rho_1 \ldots\}$ of role and agent names; $\Phi = \{\phi, \psi, \ldots\}$ of atomic propositions; $\mathcal{M} = \{act_1 \ldots\}$ of actions. Running STSs have agents. Roles are placeholders for agents in specifications. Our definitions, including language and model, are generated from this universe of discourse. Table I describes the syntax of an STS specification, including other nonterminals (particularly Expr) that we reference below. We assume an inference relation for our logic. Specifically, for a set of expressions s and an expression e, $s \vdash e$ means that e logically follows from the s.

TABLE I
NOREST SYNTAX.

Specification	$\longrightarrow$	Assumption | Action | Norm
Assumption	$\longrightarrow$	$\phi \leftarrow$ Expr | $\neg\phi \leftarrow$ Expr
Action	$\longrightarrow$	act(Label, $\mathcal{R}$, Expr, ϕ)
Norm	$\longrightarrow$	Commitment | Authorization | Prohibition
Commitment	$\longrightarrow$	c($\mathcal{R}$, $\mathcal{R}$, Expr, Expr)
Authorization	$\longrightarrow$	a($\mathcal{R}$, $\mathcal{R}$, Expr, Expr)
Prohibition	$\longrightarrow$	p($\mathcal{R}$, $\mathcal{R}$, Expr, Expr)
Expr	$\longrightarrow$	true | ϕ | $\neg$Expr | Expr $\wedge$ Expr

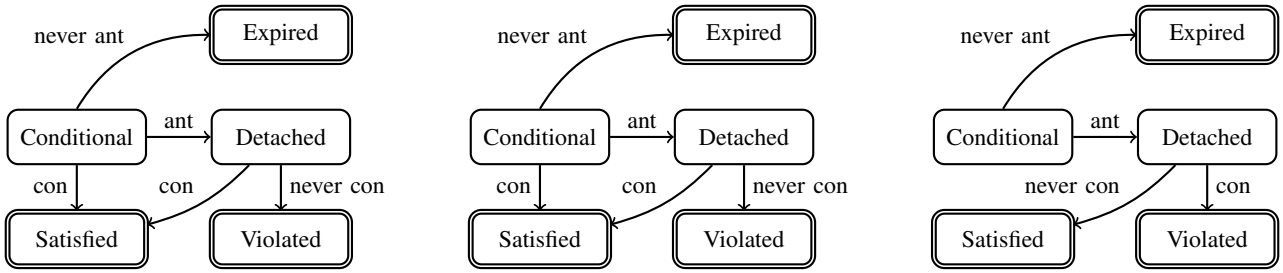
A. Norms

Definition 1 characterizes a directed norm along the lines of Singh [21], but with some improvements. We consider three types of norms: commitment, authorization, and prohibition.

Definition 1. *A norm is a tuple $\langle n, sbj, obj, ant, con\rangle$, where n, its type, is one of $\{c, a, p\}$; $sbj \in \mathcal{R}$ is its subject; $obj \in \mathcal{R}$ is its object; ant ∈ Expr is its antecedent; and con ∈ Expr is its consequent. We write a norm as n(sbj, obj, ant, con). We define $\mathcal{N}$ as the set of all norms.*

A commitment means that its subject commits to its object for bringing about the consequent if the antecedent holds. For example, EP is committed to the hospital for logging off from the ED computer after reviewing patients' EHR to prevent disclosure of their PHI [Ex5]. An authorization means that its subject is authorized by its object for bringing about the consequent if the antecedent holds. For example, EP is authorized by the hospital for accessing a patient's EHR after logging in to the ED computer [Ex2]. A prohibition means that its subject is prohibited by its object from bringing about the consequent if the antecedent holds. For example, EP is prohibited by the hospital from disclosing patients' PHI [Ex3].

Figure 2 summarizes the norm lifecycle. A commitment begins its lifecycle in the conditional state—or in detached if its antecedent is initially true. The antecedent and consequent of a conditional commitment do not hold: when the antecedent holds, the commitment becomes detached. In either case, when the consequent holds, it is satisfied. For a conditional commitment, if its antecedent becomes impossible (i.e., never ant), it expires. For a detached commitment, if the consequent becomes impossible (i.e., never con), the commitment is



(a) Commitment (accountable: subject) (b) Authorization (accountable: object) (c) Prohibition (accountable: subject)

Fig. 2. Lifecycle of norms. Double rectangles represent terminal states (i.e., the norm's lifecycle ends in those states). To highlight the differences, we use the same layout of states in each diagram.

violated. Expired, satisfied, and violated are terminal states, meaning there are no further transitions from them.

With respect to a commitment, a prohibition reverses the transitions from detached to satisfied and violated. An authorization is violated when the antecedent holds but the consequent fails to hold. A subject who is authorized for performing a consequent is not committed to doing so.

Importantly, the accountable party is the subject of a commitment or prohibition, but the object of an authorization. This understanding of authorizations as privileges of the authorized party agrees with established approaches [21], [24].

B. Assumptions

Definition 2. *A domain assumption is a pair $\langle h, b\rangle$, written $h \leftarrow b$ where h is a literal (either a member of Φ or a negation of a member of Φ) and $b \in$ Expr is any expression derived from Expr. $\mathcal{A}$ is the set of all possible assumptions.*

For example, the inference rule $\neg$logged_in $\leftarrow$ power_failure means that it is not possible to be logged in to a computer when there is a power failure. An assumption $\phi \leftarrow$ true means that ϕ must occur in every state of any enactment of the STS. We assume that there are no conflicting assumptions. That is, $\langle \neg\phi, \mathsf{true}\rangle \notin \mathcal{A}$ if $\langle \phi, \mathsf{true}\rangle \in \mathcal{A}$.

C. Actions and Regimentation

Agents' actions are supported by underlying mechanisms. An example action is accessing patient data: it is performed by EP. The mechanisms may impose *enabling conditions* upon the actions. For example, providing a token or password is an enabling condition for accessing patient data. If EP supplies the password, the access happens.

Regimentation is an implementation style in which an agent may perform an action only if that action's enabling conditions is met. Regimentation can help realize some norms through the technical architecture. Suppose an agent is authorized to perform some action given some antecedent and prohibited from performing that action when that antecedent is false. We can support these linked norms via regimentation by making the antecedent an enabling condition for the specific action. For example, suppose EP can access the patient's EHR if patient consent exists and is prohibited otherwise. Then, checking for patient consent to decide whether to allow access is a form of regimentation.

D. STS Enactments

We now describe how to operationalize an STS as a Computation Tree Logic (CTL) model [7]. Definition 3 describes an STS as consisting of sets of roles, assumptions, mechanisms, and norms. For convenience, we write instances of M as $m(\text{enabler}, \text{add}, \text{delete})$, where $m \in \mathcal{M}$, enabler $\in$ Expr, add, delete $\subseteq \Phi$ and add $\cap$ delete $= \emptyset$. When a mechanism is enabled, its effect may take place. The effect consists of a set of atomic propositions to be added and a set of atomic propositions to be deleted—these two sets are disjoint. Therefore, the effect of a mechanism is always unambiguous. A mechanism whose enabler is true is always enabled. We presume that some such mechanisms would be defined so that agents can act in the initial state.

Definition 3. *An* STS *is a tuple $\Sigma = \langle R, A, M, N\rangle$, where $R \subseteq \mathcal{R}$ is a set of roles; $A \subseteq \mathcal{A}$ is a set of domain assumptions; $M : \mathcal{M} \rightarrow$ Expr $\times$ Expr is a mapping from mechanisms to their enabling conditions and effects; and $N \subseteq \mathcal{N}$ is a set of norms.*

Definition 4 describes a CTL model. A *state* of an enactment is given precisely by the atomic propositions true therein. Below, S is interpreted as the set of possible states in the CTL model. Having a CTL formalization enables us to adopt associated formal verification tools, such as NuSMV.

Definition 4. *A CTL model is a tuple $\Gamma = \langle S, s_0, M, \delta\rangle$, where (i) $S \subseteq 2^{\Phi}$ is a subset of the powerset of Φ; (ii) $s_0 \in S$ is the initial state; (iii) $M \subseteq \mathcal{M}$ is the set of mechanisms, i.e., action labels; (iv) $\delta : S \times M \rightarrow S$ is a transition function.*

Definition 5 describes how a CTL model is obtained using the elements of an STS. The states and transitions are populated mutually inductively so as to produce a minimal CTL. (A least fixed point exists because set union is monotone.)

Definition 5. *An STS $\Sigma = \langle R, A, M, N\rangle$ generates a CTL model $\Gamma = \langle S, s_0, M, \delta\rangle$ as follows.*

- *$s_0 = \emptyset$;*
- *S is the minimal set such that $s_0 \in S$ and $(\forall s \in S : (\forall m(p, add, delete) \in M$: if $s \vdash p$, then $(s \cup add \setminus delete) \in S))$;*
- *$\delta = \{\langle s, m, s'\rangle | s, s' \in S$ and $(\exists m(p, add, delete) \in M$: $s \vdash p$ and $s' = s \cup add \setminus delete$.*

For convenience, we set the empty set as the initial state and assume interleaving of actions, that is, at most one agent acts at a time and performs at most one action. Section III describes how such a CTL model can be described for NuSMV.

E. Requirements Verification

This section develops our formal verification process. In classical RE [26], captured in Equation 1, the domain assumptions and mechanism specification together entail the requirements, meaning that under the assumptions, the specification ensures the requirements would be satisfied. Here, $\vdash$ indicates an abstract inference relation.

$$\mathrm{A}, \mathrm{M} \vdash \mathrm{R} \tag{1}$$

NoReST additionally incorporates participating agents and norms among them. An STS is correct if the domain assumptions, control mechanisms, and norms entail the requirements. Although the agents are autonomous, if they satisfy the norms, they should (cooperatively) meet the requirements. The challenge is to specify such an STS. That is, we need the following.

$$\mathrm{A}, \mathrm{M}, \mathrm{N} \vdash \mathrm{R} \tag{2}$$

Definition 6 brings out what it means to verify an STS as a way of making Equation 2 concrete.

Definition 6. *An STS Σ is* correct *with respect to requirement r if and only if Σ generates $\Gamma = \langle S, s_0, Act, \delta \rangle$ and $\Gamma \vdash r$.*

III. Norm-Based Specification

An STS specification in NoReST respects the grammar given in Table I. Listing 1 shows such a specification for controlling access to a patient's EHR in the ED. Here, EP and ED are agents, and assigned, EHR (meaning EHR is accessed) and logged_in are atomic propositions. EP is authorized to access the EHR if she is assigned to treat the patient. Moreover, EP needs to log in to ED's computer to access the EHR.

Listing 1
A specification for EP activity.

```
a(EP, ED, assigned, EHR)
act(access_EHR, EP, logged_in, EHR)
```

We apply temporal logic model checking [7] to formally verify whether an STS specification satisfies requirements. Algorithm 1 translates an STS specification (comprising propositions, roles, assumptions, actions, and norms) into an NuSMV model (comprising state transitions for each proposition).

The NuSMV syntax specifies a case statement for each variable with one or more transition rules separated by semicolons. The rule "E: Values;" means that if expression E evaluates to true, then the given variable can take any of the values in Values. For Booleans, Values can be {TRUE}, {FALSE}, or {TRUE, FALSE}. NuSMV attempts the rules within a case statement sequentially beginning from the first. Thus, a guarded transition of "TRUE: X" is executed by default and should be placed last in a case statement.

Algorithm 1 generates one case statement for each proposition $\phi \in \Phi$. For each assumption $\phi \leftarrow$ Expr, we insert a transition that forces ϕ to be true if Expr is true. Otherwise, we leave ϕ unchanged. The case of $\neg\phi \leftarrow$ Expr is similar.

Algorithm 1: T $\leftarrow$ generate(A, M, N)

Input: A: assumptions
Input: M: mechanisms
Input: N: norms
Output: T: NuSMV transition model

1 **foreach** $\phi \leftarrow$ *Expr* $\in A$ **do**
2 Add transition "Expr: TRUE;" for proposition ϕ;
3 **foreach** $\neg\phi \leftarrow$ *Expr* $\in A$ **do**
4 Add transition "Expr: FALSE;" for proposition ϕ;
5 **foreach** m*(enabler, add, delete)* $\in M$ **do**
6 Add transition "enabler: {TRUE, FALSE}; !enabler: FALSE;" for each proposition u $\in$ add;
7 Add transition "enabler: FALSE;" for each proposition v $\in$ delete;
8 **foreach** $c_i(\mathcal{R}_1, \mathcal{R}_2, Expr_1, Expr_2) \in N$ **do**
9 Add propositions det_c$_i$ and sat_c$_i$;
10 Add transition "Expr$_1$: TRUE;" for det_c$_i$;
11 Add transition "Expr$_2$: TRUE;" for sat_c$_i$;
12 Add fairness constraint "FAIRNESS !det_c$_i$ | sat_c$_i$";
13 **foreach** $a_j(\mathcal{R}_1, \mathcal{R}_2, Expr_1, Expr_2) \in N$ **do**
14 Add proposition sat_a$_j$;
15 Add transition "Expr$_2$: TRUE;" for sat_a$_j$;
16 Add fairness constraint "FAIRNESS sat_a$_j$";
17 **foreach** $p(\mathcal{R}_1, \mathcal{R}_2, Expr_1, Expr_2) \in N$ **do**
18 Add transition "Expr$_1$: FALSE;" for each proposition ϕ where $\phi \vdash$ Expr$_2$;
19 T $\leftarrow$ " ";
20 Group transitions for ϕ into [transitions(ϕ)], and add "next(ϕ):= case [transitions(ϕ)] esac;" to T;
21 **return** T;

For each mechanism m(enabler, add, delete), we insert a transition for each u in m's add list, so that u may occur when its enabling condition is true (Line 6). Similarly, we insert a transition for each v in m's delete list so as to prevent it from occurring (Line 7).

For each commitment c$_i$, we add two propositions (Line 9): det_c$_i$ to denote that c$_i$ is detached, and sat_c$_i$ to denote that c$_i$ is satisfied. The transition rules for these propositions are described as follows: det_c$_i$ is true if Expr$_1$ is true (Line 10), and sat_c$_i$ is true if Expr$_2$ is true (Line 11). A fairness constraint is added to ensure that c$_i$ is eventually satisfied when it is detached (Line 12).

For each authorization a$_j$, we add a proposition sat_a$_j$ to denote that a$_j$ is satisfied (Line 14). The transition rule for sat_a$_j$ is given (Line 15) similar to sat_c$_i$ (see Figure 2). A fairness constraint is added to ensure that a$_j$ is eventually satisfied (Line 16).

For each prohibition, we insert a transition so that the consequent will be false when the antecedent is true (Line 18), to prevent violation of the prohibition.

Finally, we group transitions for each proposition and place them within a single case statement (Line 20). Listing 2 shows the resulting NuSMV specification for a commitment. In an emergency, EP is committed to treating and operating upon patients (Line 1). This commitment is detached when emergency is true (Lines 2–5). It is satisfied when treated and operated are both true (Lines 7–10). The fairness constraint ensures that the commitment is eventually satisfied after it is detached (Lines 12–13).

Listing 2
NUSMV TRANSITIONS GENERATED FOR A COMMITMENT.

```
1  --c1(EP, ED, emergency, treated ∧ operated)
2  next(det_c1):=
3   case
4    emergency: TRUE;
5   esac;

7  next(sat_c1):=
8   case
9    treated & operated: TRUE;
10  esac;

12 FAIRNESS
13  !det_c1 | sat_c1
```

IV. NORMATIVE REFINEMENT

More than merely verifying that an STS satisfies requirements, we would like to specify an STS to ensure it satisfies the (possibly changing) stakeholder requirements (Q_2). Accordingly, we propose a refinement procedure based on normative patterns. That is, if an STS is not correct with respect to some requirements (Equation 2), then our refinement procedure computes $M' \cup N'$ such that

$$\mathrm{A}, \mathrm{M}', \mathrm{N}' \vdash \mathrm{R} \tag{3}$$

A. Norm Strength

Norm specifications can initially be broad or narrow. That is, they may not cover specific situations that might lead to opportunities being missed, and eventually cause the violation of some of the requirements. When this is the case, the stakeholders should refine the norms to account for the situation at hand. In order to come up with such a refinement, we propose to use normative patterns. Chopra and Singh [5] propose a strength relation to compare commitments such that one can be replaced with another. We extend their definition to all norm types. Having a formal means of comparing norms enables us to refine them, similar to classical refinement [19] for weakening or strengthening of inputs and outputs of methods. (Below, we use subscripts on a, c, and p as identifiers.)

Definition 7. *c($\mathcal{R}_1$, $\mathcal{R}_2$, r, u) is stronger than c($\mathcal{R}_1$, $\mathcal{R}_2$, s, v) if and only if $s \vdash r$ and $u \vdash v$.*

A commitment becomes stronger as its antecedent becomes more general and its consequent becomes more specific. That is, a stronger commitment either allows fewer alternatives to fulfill the consequent, or requires greater effort. Consider c_1(EP, ED, true, operation $\wedge$ clinic) and c_2(EP, ED, emergency, operation). Then, c_1 is stronger than c_2 because emergency $\vdash$ true and operation $\wedge$ clinic $\vdash$ operation. This is a refinement of c_1 into c_2 by weakening both the antecedent (making it more specific) and the consequent (making it more general).

Definition 8. *a($\mathcal{R}_1$, $\mathcal{R}_2$, r, u) is stronger than a($\mathcal{R}_1$, $\mathcal{R}_2$, s, v) if and only if $s \vdash r$ and $v \vdash u$.*

An authorization is stronger if its antecedent and consequent are weaker. That is, a stronger authorization allows improved flexibility. Consider a_1(EP, ED, true, own_patients $\vee$ other_patients) and a_2(EP, ED, true, own_patients). Then, a_1 is stronger than a_2 because their antecedents are identical and own_patients $\vdash$ own_patients $\vee$ other_patients.

Definition 9. *p($\mathcal{R}_1$, $\mathcal{R}_2$, r, u) is stronger than p($\mathcal{R}_1$, $\mathcal{R}_2$, s, v) if and only if $s \vdash r$ and $v \vdash u$.*

A prohibition is stronger (stricter) if its antecedent and consequent are weaker. Consider p_1(EP, ED, true, disclosed_PHI_family $\vee$ disclosed_PHI_online) and p_2(EP, ED, emergency, disclosed_PHI_online). Then, p_1 is stronger than p_2 because disclosed_PHI_online $\vdash$ disclosed_PHI_family $\vee$ disclosed_PHI_online and emergency $\vdash$ true.

B. Patterns

Table II summarizes the patterns that provide ways of extending a given norm specification with specific conditions. The relaxation patterns are inspired by healthcare law [10], and focus on modifying antecedents and consequents of norms to improve agents' flexibility, thus promoting collaboration.

Release of Liability replaces a commitment c_i with c_j where c_i is stronger than c_j.

Expansion replaces an authorization a_i with a_j where a_j is stronger than a_i.

Accessibility replaces a prohibition p_i with p_j where p_i is stronger than p_j.

The relaxation patterns of Table II expand available functionality, but can introduce vulnerabilities. Inspired by sociolegal aspects of healthcare [9], the amendment patterns address security and privacy concerns while promoting collaboration.

Responsibility replaces an authorization a_i($\mathcal{R}_1$,$\mathcal{R}_2$, r, u) with a_j($\mathcal{R}_1$,$\mathcal{R}_2$, s, v), where a_j is stronger than a_i, and adds a commitment c_k($\mathcal{R}_1$,$\mathcal{R}_2$, v, w). That is, Responsibility limits the subject of the norm only to the intended functionality provided by the relaxation pattern by specifying a complementary commitment. When the additional functionality provided by relaxing an authorization is eventually achieved by the subject, the subject commits to bringing about another proposition, so that the additional functionality is not available any more.

The Responsibility pattern turns technical requirements into social expectations by manipulating norms in the STS. Consider the following scenario: EP is authorized to access a patient's EHR from one of the desktop computers that implement an automatic session termination after inactivity. To address the need for efficiency, the ED authorizes EP to

TABLE II
REFINEMENT PATTERNS. RELAXATION PROVIDES NORM SUBSTITUTION. AMENDMENT ENSURES THE RELAXED NORM IS ACCOUNTED FOR. ENABLER PROVIDES MECHANISM UPDATE.

Pattern	Original	Refined	Conditions
Relaxation			
Release of Liability	c($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)	c($\mathcal{R}_1$,$\mathcal{R}_2$, s, v)	s⊢r, u⊢v
Expansion	a($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)	a($\mathcal{R}_1$,$\mathcal{R}_2$, s, v)	s⊢r, u⊢v
Accessibility	p($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)	p($\mathcal{R}_1$,$\mathcal{R}_2$, s, v)	s⊢r, v⊢u
Amendment			
Responsibility	a($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)	a($\mathcal{R}_1$,$\mathcal{R}_2$, s, v) c($\mathcal{R}_1$,$\mathcal{R}_2$, v, w)	s⊢r, u⊢v
Limitation	p($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)	p($\mathcal{R}_1$,$\mathcal{R}_2$, s, v) p($\mathcal{R}_1$,$\mathcal{R}_2$, v, w)	s⊢r, v⊢u
Enabler	m(r,u,v)	m(s,u,v)	r⊢s

use her phone to access EHR. However, doing so creates a vulnerability if EP leaves her phone unattended and forgets to lock it. Therefore, she has to commit to locking her phone to be granted this additional authorization. EP is accountable in case of a security breach she causes by not locking her phone.

Limitation replaces a prohibition $p_i(\mathcal{R}_1,\mathcal{R}_2, r, u)$ with $p_j(\mathcal{R}_1,\mathcal{R}_2, s, v)$, where p_i is stronger than p_j, and adds a prohibition $p_k(\mathcal{R}_1,\mathcal{R}_2, v, w)$. That is, Limitation limits the subject of the relaxed norm by specifying a complementary prohibition. When the subject is no longer prohibited from performing the consequent in certain situations, another prohibition prevents the subject from performing additional functionality.

Enabler replaces mechanism m(r,u,v) with m(s,u,v) if r ⊢ s.

The Enabler pattern refines a mechanism to relax an existing enabling condition for an action. Consider m_1(consent,non_patient_EHR,{ }). The hospital's software allows EP to access the EHR of a patient the EP is not treating only if the patient has consented. We can refine this mechanism into m_2(consent ∨ emergency,non_patient_EHR,{ }), so that the software in ED allows EP to access any patient's EHR.

C. Generating Revised Specifications

Algorithm 2 describes refinement by applying our patterns for generating refined norm specifications to satisfy requirements (Equation 3). We assume that there are no conflicting requirements. It takes as input the STS elements, a requirement to be satisfied, and refinement patterns. Its output is a set of refined norms. Note that the algorithm may return empty if there are no such norms. After the output set is initialized (Line 1), as the first step, candidate sets of norms N_{cand} are generated by applying each pattern p to each norm n in N using Φ (apply_pattern procedure in Line 4). Assume n is an authorization. If p is Release of Liability, then $N_{cand} = \emptyset$. If p is Expansion, then N_{cand} is populated with refinements of n that are constructed by adding (removing) a proposition to (from) the antecedent or consequent of n. Consider a($\mathcal{R}_1$,$\mathcal{R}_2$, r ∧ s, u). Then, N_{cand} = {a($\mathcal{R}_1$,$\mathcal{R}_2$, r, u)} ∪ {a($\mathcal{R}_1$,$\mathcal{R}_2$, s, u)} ∪ {a($\mathcal{R}_1$,$\mathcal{R}_2$, (r ∧ s) ∨ w, u)| w ∈ Φ \ {r, s, u}} ∪ {a($\mathcal{R}_1$,$\mathcal{R}_2$, r ∧ s, u ∨ v)| v ∈ Φ \ {r, s, u}}. The next step is to verify each set of norms (amendment patterns produce two norms) in N_{cand} via NuSMV. If a candidate set of norms satisfies r, it is added to the result N′ (Lines 6–7).

Algorithm 2: N′ ← refine(A, M, N, r, $\mathcal{P}$)

Input: A: assumptions
Input: M: mechanisms
Input: N: norms
Input: r: requirement
Input: $\mathcal{P}$: refinement patterns
Output: N′: refined norms

```
1 N′ ← ∅;
2 foreach n ∈ N do
3     foreach p ∈ P do
4         N_cand ← apply_pattern(n, p, Φ);
5         foreach n_cand ∈ N_cand do
6             if A, M, n_cand ⊢ r then
7                 N′ ← N′ ∪ {n_cand};
8 return N′;
```

V. HEALTHCARE PRIVACY SCENARIO

In order to demonstrate how refinement works in NOREST, we formalize the scenario described in Example 1 via norms. Table III shows the norms with a correspondence to the clause number from Example 1. Authorizations [Ex1] and [Ex2] together with commitment [Ex5] implement the security requirement for HIPAA. EP first needs to authenticate herself to enter the ED. Then, she logs in into the computer using her credentials. EP commits to logging off from the computer after she finishes reviewing EHR. Prohibition [Ex4] implements a privacy requirement that EP should only access her patient's EHR. Prohibition [Ex3] ensures that EP does not the publish patient's PHI online even in disaster situations, regarding the HIPAA privacy requirements [12]. However, in order to promote public safety, it does not prohibit EP from sharing the PHI with the patient's family, thus there is no prohibition specified [Ex6].

TABLE III
NORMS FOR EXAMPLE 1.

Label	Norm
Ex1	a(EP, ED, swiped_card, accessed_PC)
Ex2	a(EP, ED, consent ∨ logged_in, EHR)
Ex3	p(EP, ED, EHR, disclosed_PHI_online)
Ex4	p(EP, ED, logged_in, nonpatient_EHR)
Ex5	c(EP, ED, EHR, ¬logged_in)

We adopt Computation Tree Logic (CTL) to represent requirements. CTL [7] is a branching time logic, in which each branch corresponds to a possible future. Consider the

following HIPAA requirements[1]:

R-Disclose: patient's PHI must not be disclosed [11], i.e., (in CTL terms) at any point on any branch: $\mathsf{AG}\,(\neg\mathsf{disclosed\_PHI\_online})$.

R-Access: physicians must be allowed to access EHR without consent [12]. In CTL: $\mathsf{EF}\,((\mathsf{emergency} \wedge \neg\mathsf{consent}) \wedge \mathsf{EHR})$. There must be at least one branch where EHR is accessed.

R-Logout: open sessions must be closed after reviewing EHR [3]. In CTL: $\mathsf{AG}\,(\mathsf{EHR} \rightarrow \mathsf{AF}\,\neg\mathsf{logged\_in})$. For any point when EHR is accessed, EP must log out eventually on all branches.

R-Share: in case of a national disaster, EP must be allowed to share a patient's PHI with the patient's family members [12]. In CTL: $\mathsf{AG}\,(\mathsf{disaster} \rightarrow \mathsf{EF}\,\mathsf{disclosed\_PHI\_family})$. At any point when a disaster is declared, there must be at least one branch where PHI is shared.

We do not tackle requirements elicitation. Instead, we suggest refinements to improve an STS specification.

To demonstrate NOREST, we begin from norms that reflect the common practice in emergency medicine before the HIPAA privacy rule was revised in 2003 [11]. These norms do not satisfy some of the requirements. By applying our patterns, we and end with some of the norms from Table III.

Listing 3
INITIAL NUSMV MODEL.

```
1  MODULE main

3  VAR
4   EHR: boolean;
5   sat_a1: boolean;
6   ag1: {EP, ED};
7   ...

9   a1: a(ag1, ag2, consent, EHR);
10  p1: p(ag1, ag2, EHR, disclosed_PHI);

12 ASSIGN
13  init(EHR):= FALSE;
14  init(ag1):= EP;
15  ...

17  next(sat_a1):=
18   case
19    EHR: TRUE;
20   esac;

22  next(disclosed_PHI):=
23   case
24    EHR: FALSE;--from p1--
25    !EHR: FALSE;--domain assumption--
26   esac;
27  ...

29 FAIRNESS
30  sat_a1
```

Let us begin by describing the initial specification, as in Listing 3 with most variable declarations and initial state rules omitted for brevity. The elements of a state are given as Boolean variables (Lines 4–5), as enumerations (Line 6), or as instances of norms (Lines 9–10). The initial state is described by assigning values to the variables (Lines 13–14). The satisfaction condition for authorization a_1 is given by the transition rule for proposition $\mathsf{sat\_a}_1$ (Lines 17–20). A fairness constraint is added to ensure that a_1 can eventually be satisfied (Lines 29–30). The first transition rule for disclosed_PHI originates from prohibition p_1 (Line 24), and the second rule reflects a enabling condition for the domain (Line 25), i.e., patient's PHI can only be disclosed if the EHR is accessible.

Since disclosing the patient's PHI is prohibited, requirement R-Disclose is satisfied. However, this specification does not allow any flexibility in disaster situations. Thus, R-Share is not satisfied. Moreover, access to the EHR is forbidden without the patient's consent, which makes R-Access unsatisfied. R-Logout is not satisfied either, because the initial specification does not regulate computer usage in the ED.

We run Algorithm 2 for the unsatisfied requirements: R-Access, R-Logout, and R-Share. Figure 3 shows the new specification after each refinement step. We present one solution among all possible candidate refinements computed by the apply_pattern procedure.

1) Expansion: a_1 is substituted with a stronger authorization, which improves flexibility by allowing EP to access the EHR without consent.
2) Responsibility: c_1 is added as compensation for the expanded a_1, to improve security for accessing the EHR of a patient.
3) Accessibility: p_1 and p_2 are relaxed into a weaker prohibition (leaving only p_2), which prohibits EP only from publishing the patient's PHI, and not from sharing it with the patient's family, thus improving flexibility.

Note that authorization [Ex1] and prohibition [Ex4] in Table III are not generated by Algorithm 2, since they are not relevant to the requirements. Moreover, we omit assumptions and mechanisms that are not refined in this example.

We built an interface, WICKED,[2] for NOREST. For a predefined domain with assumptions and mechanisms, a modeler captures norms and requirements. Then, WICKED generates and executes a NuSMV model, and displays the verification results. For each unsatisfied requirement, a modeler can apply a suggested pattern on the current set of norms, and WICKED displays the results of refinement.

VI. MODELER STUDY

We conducted a human subject study to evaluate the effectiveness of the refinement patterns in designing an STS. Our study was declared exempt by our university's Institutional Review Board (IRB). We selected 32 graduate computer science students as study participants, and created two groups

[1]We follow the guidelines for writing requirements as described in https://www.ietf.org/rfc/rfc2119.txt.

[2]Details of the tool can be found in http://www4.ncsu.edu/~najmeri/norest/wicked/.

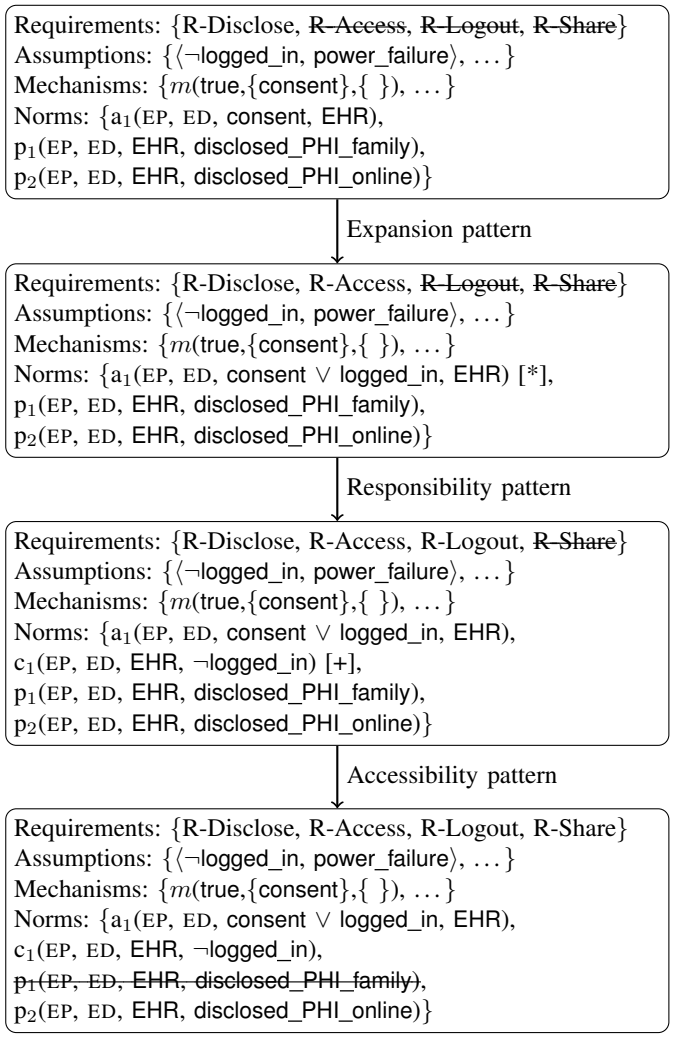


Fig. 3. Applying the patterns. The initial specification is given in the top box. Each box shows the refined specification after the application of a pattern. Crossed out requirements are not satisfied by the corresponding specification. Some norms replace existing norms [*], some are added [+], and some are removed [-].

(*Control* and NOREST) balanced in terms of familiarity with conceptual modeling and software engineering industry experience. Each participant provided informed consent; upon completion, a participant received a payment of 20 USD.

A. Study Mechanics and Deliverables

We follow a one-factor two-alternatives design. Our study has three phases in which participants work as follows.

Phase 1: Learn. Learn and define the requirements and norm specification for a healthcare privacy scenario.

Phase 2: Design. Define the requirements and norm specification for a healthcare security scenario.

Phase 3: Maintain. Comprehend and maintain the requirements and norm specification for an academic security access control scenario adopted from [22].

In each phase, participants in the NOREST group are guided with refinement patterns, whereas participants in the Control group are given a basic definition of refinement. Participants record the completion times for each phase, and complete a post-study survey at the end of the study regarding the helpfulness of patterns in their design. The first two authors designed an oracle solution for each phase, and marked the participants' designs accordingly (see Metrics below).

B. Metrics

Coverage of design: Fraction of norms in the oracle that are stated by the participants in each phase. Higher is better.

Correctness of design: Fraction of participant-stated norms that occur in the oracle for each phase. Higher is better.

Time to design: Time in minutes recorded by participants to complete each phase. Lower is better.

Ease of design: Subjective ratings provided by the participants via the post-study survey on a Likert scale (1–5, where 1 corresponds to very hard, and 5 corresponds to very easy) for each phase.

C. Hypotheses

H_1 NOREST produces specifications with greater coverage than Control.

H_2 NOREST produces specifications with greater correctness than Control.

H_3 NOREST produces specifications faster than Control.

H_4 NOREST helps produce specifications more easily than Control.

D. Results

We analyze the specifications designed by the participants in the Control and NOREST groups[3].

Observations

H_1 and H_2 are rejected for the general subject population as there are no significant differences in coverage or accuracy between NOREST and Control. We perform further analysis by grouping the subjects by their prior experience in modeling and norms. Figure 4 shows that H_1 and H_2 hold for participants with low experience in conceptual modeling or no prior knowledge of norms.

H_3 is rejected as there is no significant gain in time for the NOREST group. Further analysis suggests that for participants with no prior knowledge of norms, NOREST takes more time in the learning phase, but less time in designing and maintenance phases.

H_4 is rejected as perceived difficulty is found to be similar. However, participants in the pattern group strongly felt that patterns were helpful in their design (μ=4.4/5), and stated that an automated tool to generate alternative refinements would improve accuracy (μ=4.27/5). Moreover, both study groups employed a normative approach to capture requirements, and achieve good coverage and accuracy results (excluding minor errors regarding logic representation), which suggests that norms provide a promising way of capturing requirements. We did not employ an automated tool for NOREST, because we wanted to evaluate the usefulness of patterns without coupling

[3]The complete set of results alongside other study documents can be found in http://www4.ncsu.edu/~najmeri/norest/study/.

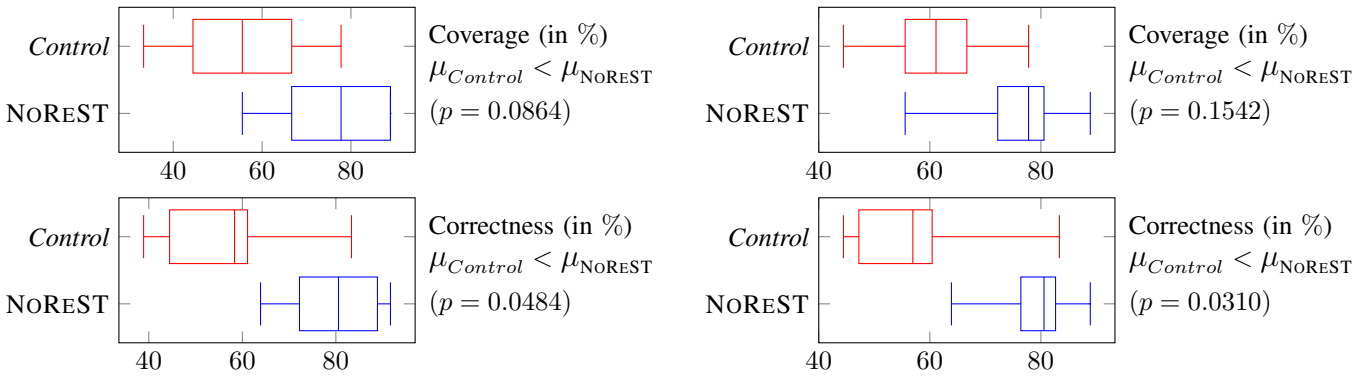


Fig. 4. Study results for coverage and correctness.

it with the effect of using a tool. For future work, we will conduct a follow-up study to see the effect of having an automated tool for refinement.

VII. Threats to Validity

The use of a formal language (such as NuSMV and CTL) has advantages for verification. However, CTL has expressiveness limitations for absolute time.

We have developed generic patterns, and although we have validated their usability, we have not investigated their completeness. Additional domain-specific patterns might be designed to better reflect some requirements.

Study Threats and Mitigation

Skill differences: We surveyed participants' about their educational backgrounds, their prior experience with conceptual modeling and software engineering industry experience. We balanced Control and NoReST with respect to aggregate experience of participants based on this survey.

Failure to report information: Participants completed the time and difficulty survey after each phase, while it was fresh in their minds.

VIII. Related Work

We discuss some leading relevant approaches in groups and show how they compare with NoReST.

Letier and Heaven [15] propose a requirements modeling approach based on events and state propositions (fluents) that synthesizes deontic automata that satisfy each individual requirement. Degiovanni et al. [8] propose a method to operationalize goals in control systems. They verify via model check whether the operations of a system are compliant with its adopted goals, and propose refinement, in the form of weakening and strengthening of the triggering conditions of operations, based on the model checker's counterexamples. Alexander et al. [1] propose a two-phase access control model for the verification of security properties in a collection of virtual machines.

The above approaches suffer from the fact that they are only applicable to control systems, and do not incorporate social aspects. The most advanced of these, Letier and Heaven's approach, involves permissions and obligations that are superficially similar to authorizations and commitments, they do not accommodate autonomy, and instead assume obligations are assigned to component, which complies with them. In contrast, NoReST provides a richer specification with the addition of autonomous agents and social norms. Agents bring autonomy, and norms provide accountability for their (unpredictable) actions in an open system.

Barth et al. [2] formalize the norms of transmitting personal information in temporal logic, and discuss properties such as consistency, entailment, and compliance. They study HIPAA scenarios similar to ours. Like Letier and Heaven, Barth et al. make strong assumptions on agent autonomy (i.e., agents never violate norms), and they do not discuss how agents interact with nonautonomous components. NoReST regulates such interactions using a richer normative formalization. Moreover, we provide refinement patterns based on the underlying normative theory, and validate their usability.

Model checking has been employed before for verification of privacy policies. Kafalı et al. [14] verify whether shared information propagates as expected among the users of a social network. However, they do not tackle formal normative models or provide refinement when a privacy policy is violated.

Other approaches tackle goal refinement. Siena et al. [20] derive and refine goals to comply with the norms. Horkoff et al. [13] discuss goal refinement in the context of business intelligence modeling. We do not address how agents adopt goals, but focus on normative refinement so as to allow more flexibility for the agents when they are pursuing their stakeholders' requirements.

Tun et al. [23] follow Barth et al.'s notion of privacy [2], and introduce privacy arguments to analyze privacy preferences of users. They reason about privacy via *decreasoner*, an Event Calculus based tool, and present a case study using the *Buddy-Tracker* mobile application. Tun et al.'s approach accounts for agent autonomy, however, it can only identify norm violations, whereas NoReST additionally suggests refinement in case of violations. Omoronyia et al. [17] present Caprice, a tool based on Tun et al., to design adaptive applications. Caprice uses the notion of privacy awareness requirements to identify

attributes that could lead to privacy threats, and identifies contexts that can violate privacy. Based on the severity of a privacy threat, Caprice recommends a mitigation action determined using predefined adaption rules. Yu et al. [25] explore the risks associated with security requirements, and propose automated checking of arguments for assessing the security of a system. Whereas these approaches detect possible privacy and security violations, they do not support a design-time refinement process as we do when some requirements are not satisfied.

Peng et al. [18] propose an approach for the adaptation of agent behaviors (via plan updates) at run-time due to changing commitments in open, dynamic STSs. In contrast, NOREST does not focus on the internal reasoning of agents, but instead refines the norms to ensure correct behavior. Li et al. [16] formalize ambiguous stakeholders requirements via refinement operators to specialize generic requirements. However, their modeling language does not adopt a normative approach, and thus cannot handle autonomous interactions.

IX. CONCLUSIONS

We proposed NOREST, a formal model for STSs based on norms that supports requirements verification via model checking and refinement via normative patterns. We conducted an empirical study, and showed that a design process with NOREST's patterns is helpful for subjects with limited experience in conceptual modeling and norms.

Our work opens up interesting directions for enhancements. Whereas our norm refinement patterns are generic, it will help to investigate domain-specific patterns (e.g., switching to priority norms for life-threatening situations). Moreover, NOREST can be extended with probabilistic reasoning, and realized in a probabilistic model checker such as PRISM. This will enable the investigation of more realistic scenarios where deterministic reasoning is not adequate, e.g., how likely is a norm to be violated given an STS specification.

ACKNOWLEDGMENTS

This research is supported by the US Department of Defense under the Science of Security Lablet grant.

REFERENCES

[1] P. Alexander, L. Pike, P. Loscocco, and G. Coker. Model checking distributed mandatory access control policies. *ACM Transactions on Information and System Security*, 18(2):6:1–6:25, July 2015.

[2] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 184–198, Washington, DC, 2006. IEEE Computer Society.

[3] Y. B. Choi, K. E. Capitan, J. S. Krause, and M. M. Streeper. Challenges associated with privacy in health care industry: Implementation of HIPAA and the security rules. *Journal of Medical Systems*, 30(1):57–64, 2006.

[4] A. K. Chopra, F. Dalpiaz, F. B. Aydemir, P. Giorgini, J. Mylopoulos, and M. P. Singh. Protos: Foundations for engineering innovative sociotechnical systems. In *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE)*, pages 53–62, Karlskrona, Aug. 2014. IEEE Computer Society.

[5] A. K. Chopra and M. P. Singh. Multiagent commitment alignment. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 937–944, 2009.

[6] A. K. Chopra and M. P. Singh. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *Proceedings of the 25th International World Wide Web Conference*, pages 1–12, Montréal, Apr. 2016. ACM.

[7] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.

[8] R. Degiovanni, D. Alrajeh, N. Aguirre, and S. Uchitel. Automated goal operationalisation based on interpolation and SAT solving. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 129–139, 2014.

[9] C. Ham, R. Dingwall, P. Fenn, and D. Harris. *Medical Negligence: Compensation and Accountability*. Centre for Socio-Legal Studies, King's Fund Institute, London, 1988.

[10] D. Hammaker. *Health Care Management and the Law: Principles and Applications*. Cengage Learning, Boston, 2010.

[11] HHS. Summary of the HIPAA privacy rule. United States Department of Health & Human Services (HHS), 2003. http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/.

[12] HHS. Bulletin: HIPAA privacy in emergency situations. United States Department of Health & Human Services (HHS), 2014. http://www.hhs.gov/ocr/privacy/hipaa/understanding/special/emergency/.

[13] J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos. Strategic business modeling: Representation and reasoning. *Software and Systems Modeling*, 13(3):1015–1041, July 2014.

[14] Ö. Kafalı, A. Günay, and P. Yolum. Detecting and predicting privacy violations in online social networks. *Distributed and Parallel Databases*, 32(1):161–190, Mar 2014.

[15] E. Letier and W. Heaven. Requirements modelling by synthesis of deontic input-output automata. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pages 592–601, 2013.

[16] F.-L. Li, J. Horkoff, A. Borgida, G. Guizzardi, L. Liu, and J. Mylopoulos. From stakeholder requirements to formal specifications through refinement. In S. A. Fricker and K. Schneider, editors, *Requirements Engineering: Foundation for Software Quality*, volume 9013 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2015.

[17] I. Omoronyia, L. Pasquale, M. Salehie, L. Cavallaro, G. Doherty, and B. Nuseibeh. Caprice: A tool for engineering adaptive privacy. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 354–357, Essen, Germany, 2012. ACM.

[18] X. Peng, Y. Xie, Y. Yu, J. Mylopoulos, and W. Zhao. Evolving commitments for self-adaptive socio-technical systems. In *Proceedings of the 19th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 98–107, Aug 2014.

[19] K. Pohl. The three dimensions of requirements engineering: A framework and its applications. *Information Systems*, 19(3):243–258, Apr. 1994.

[20] A. Siena, S. Ingolfo, A. Susi, I. J. Jureta, A. Perini, and J. Mylopoulos. Requirements, intentions, goals and applicable norms. In S. Castano, P. Vassiliadis, L. V. Lakshmanan, and M. L. Lee, editors, *Advances in Conceptual Modeling*, volume 7518 of *Lecture Notes in Computer Science*, pages 195–200. Springer Berlin Heidelberg, 2012.

[21] M. P. Singh. Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):21:1–21:23, Dec. 2013.

[22] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh. Engineering topology aware adaptive security: Preventing requirements violations at runtime. In *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*, pages 203–212, 2014.

[23] T. T. Tun, A. Bandara, B. Price, Y. Yu, C. Haley, I. Omoronyia, and B. Nuseibeh. Privacy arguments: Analysing selective disclosure requirements for mobile applications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE)*, pages 131–140, Chicago, 2012.

[24] G. H. Von Wright. Deontic logic: A personal view. *Ratio Juris*, 12(1):26–38, 1999.

[25] Y. Yu, V. N. L. Franqueira, T. Than Tun, R. J. Wieringa, and B. Nuseibeh. Automated analysis of security requirements through risk-based argumentation. *Journal of Systems and Software*, 106(C):102–116, Aug. 2015.

[26] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1):1–30, Jan 1997.