Desen: Specification of Sociotechnical Systems via Patterns of Regulation and Control

ÖZGÜR KAFALI, University of Kent
NIRAV AJMERI, North Carolina State University
MUNINDAR P. SINGH, North Carolina State University

We address the problem of engineering a sociotechnical system (STS) with respect to its stakeholders' requirements. We motivate a two-tier STS conception comprising a technical tier that provides control mechanisms and describes what actions are allowed by the software components, and a social tier that characterizes the stakeholders' expectations of each other in terms of norms. We adopt agents as computational entities, each representing a different stakeholder. Unlike previous approaches, our framework, Desen, incorporates the social dimension into the formal verification process. Thus, Desen supports agents potentially violating applicable norms—a consequence of their autonomy. In addition to requirements verification, Desen supports refinement of STS specifications via design patterns to meet stated requirements. We evaluate Desen at three levels. We illustrate how Desen carries out refinement via the application of patterns on a hospital emergency scenario. We show via a human-subject study that a design process based on our patterns is helpful for participants who are inexperienced in conceptual modeling and norms. We provide an agent-based environment to simulate the hospital emergency scenario to compare STS specifications (including participant solutions from the human-subject study) with metrics indicating social welfare and norm compliance, and other domain dependent metrics.

CCS Concepts: • **Security and privacy** → **Security requirements**; **Human and societal aspects of security and privacy**; • **Computing methodologies** → **Knowledge representation and reasoning**; **Multi-agent systems**; • **Theory of computation** → *Verification by model checking*;

Additional Key Words and Phrases: Agent-oriented software engineering, norms, security and privacy requirements, design patterns, simulation

ACM Reference format:
Özgür Kafalı, Nirav Ajmeri, and Munindar P. Singh. 2019. Desen: Specification of Sociotechnical Systems via Patterns of Regulation and Control. *ACM Trans. Softw. Eng. Methodol.* 1, 1, Article 1 (October 2019), 54 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

A sociotechnical system (STS) is composed of both social (people and organizations) and technical (computers and networks) elements [Dalpiaz et al. 2013; Kafalı et al. 2016a; Sommerville et al. 2012]. In this conception, we understand an STS as a normative multiagent system (nMAS), wherein autonomous agents representing stakeholders interact with each other through and about the technical components [Chopra et al. 2014; Singh 2013]. Accordingly, an nMAS supports interaction across two levels: (i) among the agents regarding the applicable social norms, and (ii) between the

1049-331X/2019/10-ART1 $15.00
https://doi.org/0000001.0000001

agents and the technical components that belong to, and are subject to control by, the underlying technical architecture.

Figure 1 illustrates this conception. We place norms [Barth et al. 2006; Singh 2013] as central to nMAS. A norm here is a directed relationship between two agents. An nMAS's technical tier comprises (nonautonomous) software components supporting various agent actions. Multiple agents can use a software component concurrently, and an agent can use multiple software components. Access to software components is controlled via technical mechanisms, which embody hard specifications and allow or prevent specific agent actions. In essence, we understand such mechanisms as providing *regimentation* [Jones and Sergot 1993]. An nMAS's social tier comprises autonomous agents interacting with each other. Norms *regulate* [Jones and Sergot 1993; Kafalı et al. 2016a; Singh 2013] such interactions by providing accountability for agent actions. That is, agents, being autonomous parties, can violate their norms but each norm specifies who is accountable to whom for what and when. Since norms can be violated, legal compliance is not assured in general. However, as appropriate, we can ensure compliance for some norms via regimentation and can achieve regulation through social controls such as sanctions [Nardin et al. 2016].

An nMAS designer specifies the technical tier (mechanisms) and social tier (norms) according to the requirements of its stakeholders. A well-conceived nMAS would capture ways in which to recover from violations, e.g., by sanctioning an accountable party for a violation. For target applications of nMAS, such as healthcare security and privacy, incorporating norms is essential to gain the flexibility needed to satisfy stakeholder requirements that would otherwise be left unsupported.

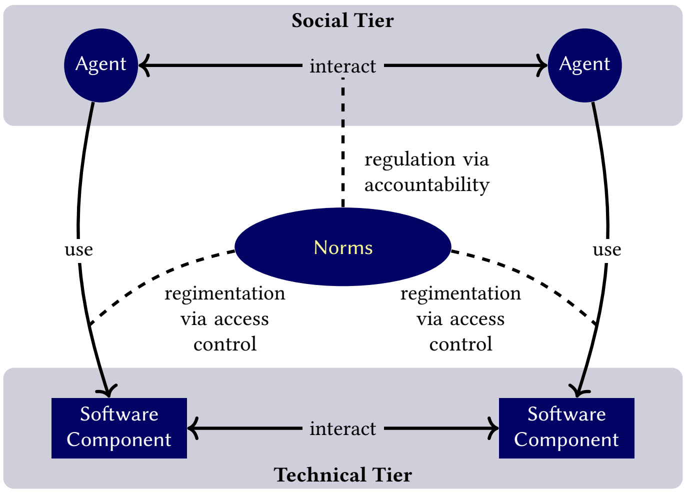


Fig. 1. Two-tier normative multiagent system (nMAS) with n ($\geqslant 2$) agents and m ($\geqslant 2$) software components. Solid arrows represent how (autonomous and nonautonomous) entities interact with each other. Dashed lines represent how norms provide regulation and regimentation.

Running Example: HIPAA (US Health Insurance Portability and Accountability Act) Emergency Rule [HHS 2014]. The following nMAS supports regulations regarding the disclosure of patient information in national emergencies. For ease of presentation, we adopt distinct type styles for AGENTS and propositions.

Example 1. The guidelines for disasters provided by the American College of Emergency Physicians, ACEP [ACEP 2013], include expanding staff capacity and relaxing privacy requirements. Following such guidelines, a hospital may recruit outside physicians to cope with the load of a national emergency or disaster situation. However, doing so inevitably raises privacy concerns regarding the patients' electronic health records (EHR). According to HIPAA privacy rules, during regular medical practice, physicians are required to obtain consent from a patient to share the patient's protected health information (PHI) with the patient's family members. In addition, patients can request confidential communications, which would prohibit a physician from consulting another colleague about the patient's condition. However, in emergencies, a patient is in no condition to give consent, and the HIPAA rules are waived to authorize physicians to inform a patient's family about the condition of the patient. In case a hospital needs to recruit outside physicians, the hospital must prohibit the outside physicians from disclosing patients' PHI.

We now informally apply Figure 1's conception to Example 1. Figure 2 instantiates our conceptual model with agents, norms, and software components. The agents are PHYSICIAN, OUTSIDE_PHYSICIAN, HOSPITAL, PATIENT, and FAMILY. The hospital's EHR software implements access control mechanisms (corresponding to a software component in the figure), where the credentials are provided by HOSPITAL to the staff based on their roles. For example, a mechanism enables PHYSICIAN to access a patient's EHR only when there is consent from PATIENT (corresponding to a *use* relation in the figure). However, the same mechanism enables access to all emergency patients' EHRs when a national emergency is declared for the geographical area that the hospital serves. Once physicians have access to a patient's EHRs, a technical mechanism cannot prevent them from disclosing the patient's PHI. Therefore, in the social tier, HOSPITAL prohibits PHYSICIAN from disclosing a patient's PHI (corresponding to a norm in the figure). However, PHYSICIAN may share a patient's PHI with OUTSIDE_PHYSICIAN in a national emergency. OUTSIDE_PHYSICIAN may then inform FAMILY about the patient's medical situation.

The potential benefits of a normative approach are significant. First, it captures the social and technical elements together, whereas traditional approaches artificially separate them. Second, a normative approach formally represents and reasons about the social tier. Investigating the two tiers together brings an important research challenge to fore: What computational representations and techniques can help realize this vision? We refine this challenge into two major research questions that this paper addresses by emphasizing a sociotechnical perspective on nMAS.

RQ_1: Verification. How can we verify that an STS, via its technical and social elements, satisfies the requirements of its stakeholders?
Significance: Situations in which agents stand in for autonomous stakeholders are especially crucial to security and privacy [Ajmeri et al. 2018; Kökciyan and Yolum 2016; Such and Criado 2016], because security and privacy concerns presuppose that multiple autonomous agents are involved.
Novelty: Previous research on verification of software systems falls into several bodies. Formal methods such as Degiovanni et al. [2014] and Letier and Heaven [2013] for modeling and synthesis of software components do not incorporate autonomous agents into their architectures. Works that incorporate agents such as Barth et al. [2006] artificially limit the agents' autonomy and assume they always comply with any applicable norms. Similarly, multiagent modeling and verification approaches such as Dastani et al. [2010], Kardas [2013], and Kuster et al. [2014] do not investigate nonfunctional (e.g., security and privacy) stakeholder requirements in depth. Agent-oriented software engineering (AOSE) methodologies such as Abushark et al. [2015], Bresciani et al. [2004], Chopra and Singh [2016b], Gómez-Sanz and Fuentes-Fernández [2015], Horkoff and Yu [2016], Parunak and Brueckner [2015] focus on modeling techniques to aid software engineering by

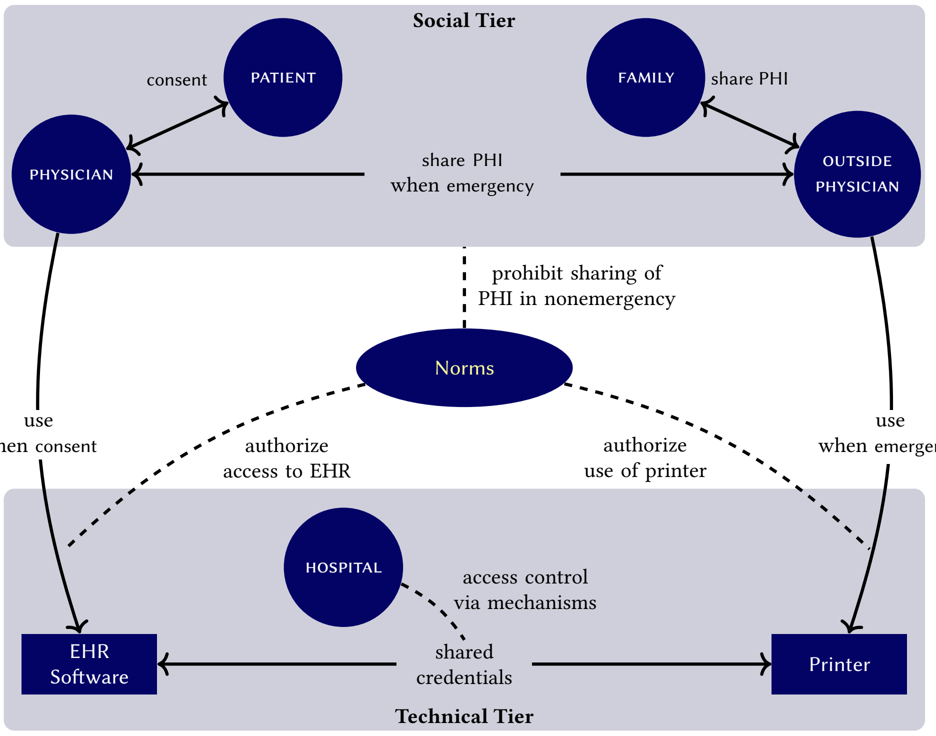


Fig. 2. Instance of an nMAS with agents, norms, and software components.

integrating autonomous components in the software development lifecycle. However, they do not incorporate the social dimension.

RQ_2: Refinement. How can we guide the refinement of an STS specification to ensure that it satisfies the stated and changing requirements?
Significance: Creating specifications is at the heart of software engineering; doing so for STSs with first-class status for the social tier helps avoid loss of user requirements via ad hoc translation from the social tier to the technical tier. Accommodating requirement changes in an STS is nontrivial.
Novelty: Existing approaches on STSs, e.g., Protos [Chopra et al. 2014], provide an abstract design process but not concrete support for producing specifications. Formal requirements engineering approaches investigate privacy policies [Bhatia et al. 2016], regulations [Gandhi and Lee 2011], and breach reports [Kafalı et al. 2017b]. However, they do not discuss how such investigation can guide the refinement of an STS specification. We use agent-based simulation to evaluate candidate STS designs and guide their refinement.

Contributions and organization. We propose DESEN (Turkish for pattern), a formal framework for the verification and refinement of STS specifications via social norms (Section 2). DESEN supports formal verification of requirements by generating a temporal model based on the correct behavior described by the norms (Section 3), thus contributing to security requirements engineering with sociotechnical considerations [Dalpiaz et al. 2016]. In case an STS specification does not satisfy a requirement, DESEN supports refinement based on design patterns (Section 4). DESEN is novel because it provides a way to incorporate STS considerations in AOSE via design patterns, which have not been explored in depth by previous work, including our previous research [Kafalı et al.

2016a,b]. Our extensive set of patterns transforms STS specifications between the technical and social tiers, thereby providing STS designers alternative means to satisfy stakeholder requirements. We evaluate Desen at three levels. We demonstrate how refinement works on a national emergency scenario from HIPAA (Section 5), which covers the legal and privacy requirements of HIPAA pertaining to emergencies. We conduct a human-subject study to evaluate the effectiveness of our pattern-based approach in capturing and refining stakeholder requirements (Section 6). We provide an open-source, agent-based simulation environment to mimic a social community consisting of multiple hospitals, physicians, and patients (Section 7). Our simulation experiments compare STS designs with respect to additional requirements by computing generic metrics indicating social welfare and norm compliance as well as domain dependent metrics such as the number of alive residents in various community settings, e.g., extreme national emergency and regular medical practice.

Practical usage and implications. Desen provides an analyst (or a designer) with a systematic way to verify and refine STS specifications. Specifically, after the initial elicitation of requirements (not tackled here) and a set of initial designs based on individual consideration of the requirements, the analyst (i) verifies the designs with respect to the requirements using a model checker tool; (ii) assuming there are unsatisfied requirements in some of the designs – sets up the simulation environment with environment parameters obtained from the requirements and parameters related to norms and mechanism obtained from the designs; (iii) eliminates designs that do not perform well in the simulation; (iv) refines the rest of the designs by applying the patterns (see Section 7 for a sample comparison of STS designs); (v) goes back to step (i). Note that a wrapper tool would hide the technical details of the formal verification process from the analyst. However, the development of such a tool is out of the scope for this paper.

2 FORMAL FRAMEWORK

We now describe the formal constructs that make up the technical and social tiers of an nMAS. Our universe of discourse comprises three finite sets: (1) $\mathbb{R} = \{\alpha_1 \ldots\}$ of role and agent names; $\Phi = \{\phi \ldots\}$ of atomic propositions; and $\mathcal{M} = \{m_1 \ldots\}$ of mechanisms. Run-time nMASs involve agents. Roles are placeholders for agents in design-time specifications. Our definitions, including language and model, are generated from this universe of discourse. Table 1 describes the syntax of an nMAS specification, including other nonterminals (particularly Expr and List) that we reference below.

2.1 Computation Tree Logic

We operationalize an nMAS via a branching-time model of the kind used for Computation Tree Logic (CTL) [Clarke et al. 1999]. We adopt the CTL syntax used by the NuSMV model checker [Cimatti et al. 2002]. CTL introduces the temporal operators AX, AF, AG, AU, EX, EF, EG, and EU, each of which combines quantification over paths (each path being a possible branch of time) and quantification within a path. Specifically, A and E quantify over paths. A stands for *all*; that is, Aq means that the quantified formula q holds on all paths emanating from the current point. E stands for *exists*; that is, Aq means that the quantified formula q holds on at least one path emanating from the current point. And, X, F, G and U are specific to a single path. X stands for *next* and Xp means that p holds at the next state on the given path. F stands for *eventually* and Fp means that p holds eventually at some future states on the given path. G stands for *globally* and Gp means that p holds at all future states on the given path. Finally, U stands for *until* and pUq means that p holds on the given path until q holds on the given path. Note that ∪ (cup symbol) in Table 1 is the set union operator, and not to be confused with CTL until operator U (sans serif U).

Table 1. DESEN syntax.

Specification	$\longrightarrow$	{ } | {Specification} ∪ {Specification} | Norm | Assumption | Mechanism
Norm	$\longrightarrow$	Commitment | Authorization | Prohibition
Commitment	$\longrightarrow$	c($\mathbb{R}$, $\mathbb{R}$, Expr, Expr)
Authorization	$\longrightarrow$	a($\mathbb{R}$, $\mathbb{R}$, Expr, Expr)
Prohibition	$\longrightarrow$	p($\mathbb{R}$, $\mathbb{R}$, Expr, Expr)
Assumption	$\longrightarrow$	$\langle \phi, Expr \rangle$ | $\langle \neg\phi, Expr \rangle$
Mechanism	$\longrightarrow$	m(Expr, List, List)
Expr	$\longrightarrow$	true | ϕ | ¬Expr | $\overline{\text{Expr}}$ | Expr ∧ Expr
List	$\longrightarrow$	{ } | ϕ | {ϕ} ∪ {ϕ}

2.2 Events and Immutability

In our setting, agents interact with each other and the environment. The interface with the environment is in terms of observed events. In essence, the series of observations of events can only grow over time. We assume for each event ev, there exists an event that is the complement of ev, denoted by $\overline{\text{ev}}$, which describes the *nonoccurrence* of ev. That is, we formalize $\overline{\text{ev}}$ in CTL as AG ¬ev. Complement events may thus correspond to deadline conditions. For example, if operate means that the physician has operated within the specified time interval, then $\overline{\text{operate}}$ means that the physician has failed to operate upon the patient during that time interval. Note that both operate and $\overline{\text{operate}}$ cannot hold. Therefore, events are *stable*—immutable once they have occurred [Chopra and Singh 2015].

2.3 Norms and Assumptions

Definition 1 characterizes a directed norm along the lines of Singh [2013], but with some enhancements.

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

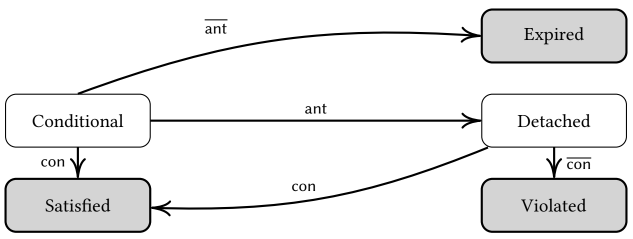


Fig. 3. Commitment lifecycle. The accountable party is the subject of the norm. White rectangles represent nonterminal states. Shaded rectangles represent terminal states.

We consider three kinds of norms. A commitment (c) means that its subject commits to its object to bringing about the consequent if the antecedent holds. Consider the following commitment:

c(PHYSICIAN, HOSPITAL, emergency, operate). A physician PHYSICIAN is practically committed to the hospital HOSPITAL to operating upon patients in an emergency. The physician is accountable to the hospital for this commitment. If the physician fails to operate upon patients, the commitment is violated. Figure 3 summarizes the norm lifecycle for a commitment. A norm begins its lifecycle in the conditional state—or in the detached state if its antecedent is initially true. When a commitment is conditional, neither the antecedent nor the consequent hold. When the antecedent holds, the commitment becomes detached. For a conditional commitment, if its antecedent condition (ant) becomes impossible to satisfy, it expires. Recall the discussion of events and complementation in Section 2.2. Expired is a terminal state, i.e., the commitment's lifecycle ends in the expired state. Note that it is possible for an antecedent such as emergency that the norm never transitions into the expired state. Moreover, a norm whose antecedent is true does not expire. However, for some antecedents, such as consent, expiration is possible (e.g., patients cannot give consent when they are in an emergency). For a detached commitment, if the consequent condition (con) is brought about, the commitment is satisfied. If the consequent becomes impossible to satisfy (i.e., $\overline{\text{con}}$), the commitment is violated. Like expired, satisfied and violated are terminal states, meaning there are no further transitions from them.

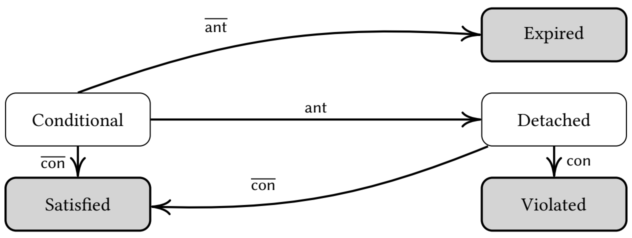


Fig. 4. Prohibition lifecycle. The accountable party is the subject of the norm. White rectangles represent nonterminal states. Shaded rectangles represent terminal states.

A prohibition (p) means that its subject is prohibited by its object from bringing about the consequent if the antecedent holds. Consider the following prohibition: *p*(PHYSICIAN, HOSPITAL, true, share_PHI_thirdparty). A physician PHYSICIAN is prohibited by the hospital HOSPITAL from disclosing a patient's PHI to others (share_PHI_thirdparty). The physician is accountable to the hospital for this prohibition. This prohibition is unconditional because its antecedent is true. If the patient's PHI is disclosed, the prohibition is violated. Figure 4 summarizes the norm lifecycle for a prohibition.

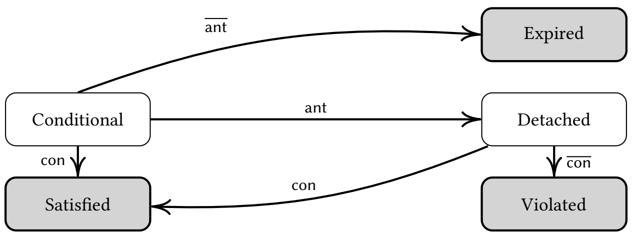


Fig. 5. Authorization lifecycle. The accountable party is the object of the norm. White rectangles represent nonterminal states. Shaded rectangles represent terminal states.

An authorization (a) means that its subject is authorized by its object for bringing about the consequent if the antecedent holds. Consider the following authorization: *a*(PHYSICIAN, HOSPITAL, consent, EHR ∨ operate). A PHYSICIAN is authorized by HOSPITAL to access a patient's EHR as well as to operate upon the patient when the patient's consent is obtained. Here, the object (HOSPITAL) is accountable to the subject (PHYSICIAN). If the physician cannot access the patient's EHR or operate upon the patient when the authorization is detached, then the authorization is violated. An authorization acts as a commitment on the authorizing party making it accountable. That is, the authorizing party should ensure the authorized party is not blocked from the consequent if the antecedent holds [Von Wright 1999]. Note that the authorization semantics captures this accountability relation, therefore an additional commitment is not required. Figure 5 summarizes the norm lifecycle for an authorization.

DEFINITION 2. *A domain assumption is a pair $\langle h, b \rangle$, 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. An assumption holds if and only if the CTL formula* $\mathrm{AG}\,(b \rightarrow h)$ *is true. $\mathcal{A}$ is the set of all possible assumptions.*

Assumptions describe domain-specific statements. For example, ⟨¬logged_in, power_failure⟩ means that physicians cannot log in to their computers when there is a power failure. Assumption $\langle \phi, \mathsf{true} \rangle$ means that ϕ must occur in every state of any enactment of the nMAS. A norm creates an expectation about its parties under the assumption that they are trustworthy. For example, the subject of a prohibition is not expected to bring about the prohibition's consequent: ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩. Trustworthiness could be per norm with a logical structure as in Singh [2011].

2.4 Mechanisms and Regimentation

Mechanisms are supported by the underlying technical architecture. An example mechanism is accessing patient data. The technical architecture may impose *enabling conditions* upon mechanisms. For example, providing a token or password is an enabling condition for accessing patient data. If PHYSICIAN supplies the password, the access happens.

Regimentation is an implementation style in which an agent may perform a mechanism only if that mechanism's enabling conditions are met. To provide such regimentation, we introduce a formalization of mechanisms into our model. We write mechanisms as m(enabler, add, delete), where enabler ∈ Expr, add, delete ⊆ Φ and add ∩ delete = { }. The underlying intuition is that when a mechanism is enabled (i.e., enabler is true), its effect may take place. The effect consists of a set of atomic propositions to be added (add list) and a set of atomic propositions to be deleted (delete list)—these two sets are disjoint. Therefore, the effect of a mechanism is always unambiguous.

Enabler: A mechanism whose enabler is true is always enabled. We presume that some such mechanisms (i.e., always enabled) would be defined so that agents can act in the initial state. The mechanism m(true, {consent}, {}) allows a patient to electronically sign consent at all times.

Add list: Consider the mechanism m(password, {logged_in}, {}), for logging in to a computer. That is, when the password is provided (i.e., enabler becomes true), this mechanism allows the user to become logged in since the proposition logged_in is contained in the add list.

Delete list: Consider a logout mechanism m(logged_in, {}, {logged_in}). When logged in, this mechanism allows the user to log out. That is, the proposition logged_in that is contained in the delete list is removed from the current state.

2.5 nMAS Enactments

We now describe how to operationalize an nMAS in terms of a branching-time model, as used for CTL. Definition 3 specifies an nMAS as consisting of sets of roles L, assumptions A, mechanisms M, and norms N (see Table 1 for the syntax).

DEFINITION 3. *An* nMAS *is a tuple* $\Sigma = \langle L, A, M, N \rangle$*, where* $L \subseteq \mathbb{R}$ *is a set of roles;* $A \subseteq \mathcal{A}$ *is a set of assumptions;* $M : \mathcal{M} \longrightarrow Expr \times 2^{\Phi} \times 2^{\Phi}$ *is a mapping from mechanisms to enabling conditions and effects; and* $N \subseteq \mathcal{N}$ *is a set of norms.*

Consider an nMAS for a hospital environment: $\Sigma_{hospital}$ = ⟨ {HOSPITAL, PHYSICIAN, PATIENT }, {⟨trustworthy(PHYSICIAN), true⟩, ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩}, {$\mathrm{m}_{consent}$(true, {consent}, {}), m_{access}(consent, {EHR}, {})}, {p_{share}(PHYSICIAN, HOSPITAL, true, share_PHI_thirdparty)}⟩. There are three agent roles: hospital, physician, and patient. There are two assumptions: physicians are trustworthy and trustworthy agents do not violate their prohibitions. There are two mechanisms: patients can give consent and physicians can access patients' EHR if consent is provided. There is one norm: physicians are prohibited by the hospital from disclosing patients' PHI.

Definition 4 specifies 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 labeled states in the CTL model. Having a CTL formalization enables us to exploit associated formal verification tools, such as NuSMV.

DEFINITION 4. *A CTL model is a tuple* $\Gamma = \langle S, s_0, M, \delta, Lbl \rangle$*, 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; (iv)* $\delta : S \times M \longrightarrow S$ *is a transition function; and (v) Lbl is a labeling function for states.*

For example, a CTL model, $\Gamma_{hospital}$, corresponding to $\Sigma_{hospital}$ would have states consisting of propositions that are contained in the add and delete lists of the mechanisms in $\Sigma_{hospital}$. Similarly, the transitions among those states are carried out according to the mechanism definitions in $\Sigma_{hospital}$, e.g., the mechanism for accessing a patient's PHI (m_{access}) can only be performed from a state that contains the proposition consent.

Definition 5 describes how a CTL model is generated from an nMAS. The states and transitions are populated mutually inductively so as to produce a minimal CTL model (a least fixed point exists because set union is monotonic).

DEFINITION 5. *An nMAS* $\Sigma = \langle L, A, M, N \rangle$ *generates a CTL model* $\Gamma = \langle S, s_0, M, \delta, Lbl \rangle$ *as follows:*

(1) s_0 *includes the set of propositions that are always true in the given domain;*
(2) 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* $((s \cup add) \setminus delete) \in S))$*;*
(3) $\delta = \{\langle s, m, s' \rangle | s, s' \in S$ *and* $m(p, add, delete) \in M$ *such that* $s \vdash p$ *and* $s' = (s \cup add) \setminus delete\}$*.*

The initial state s_0 includes only domain facts (i.e., the head of an assumption where the body is true). The rest of the states are constructed using the mechanism definitions, i.e., successor states are added using the propositions contained in the add and delete lists of mechanisms only when the enabler conditions of the mechanisms are satisfied in the predecessor state. We assume interleaving of mechanisms, that is, at most one agent acts at a time and performs at most one mechanism. This definition provides the formal basis of our correctness check for nMASs in Definition 6. Figure 6 demonstrates part of a CTL model to illustrate the connection between an nMAS and a CTL model, and how we can check the correctness of an nMAS using its CTL model. Section 3 describes how to generate such a CTL model for NuSMV.

2.6 Requirements Verification

This section develops our formal verification process. Classical requirements verification, as characterized by Zave and Jackson [1997], states the following:

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

That is, assumptions (A) and mechanisms (M) together entail requirements (R), meaning that under the stated assumptions, the specification ensures the requirements would be satisfied. In Zave and Jackson's formulation, $\vdash$ indicates a notional inference relation. Although they do not explicitly rule out norms, the software specification in their formulation is what provides the functional components (M), which leaves the only place for norms as part of the assumptions (A). In contrast, in our formalization, both technical mechanisms and social norms are captured in computational terms. Therefore, we can describe the interplay between the technical and social tiers and show how sociotechnical patterns can apply to both tiers.

Accordingly, in addition to the assumptions and mechanisms that traditional software engineering incorporates into requirements verification, Desen incorporates the participating agents and their norms (N) as part of the social tier. An nMAS is correct if assumptions, mechanisms, and norms together entail the requirements (as formalized in Definition 6). If the agents satisfy the norms, they would jointly meet the requirements provided the mechanisms are implemented and the domain assumptions hold. The challenge is to specify such an nMAS. That is, we need the following:

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

In the above equation, the $\vdash$ can be understood as a notional inference relation, the same as Zave and Jackson's, to draw out the conceptual distinction between our approach and theirs. However, our formalization makes the $\vdash$ concrete by adopting CTL as the underlying logic. That is, given an expression e and a set of expressions s, we write $s \vdash e$ to mean that for any CTL model Γ and state i in Γ, if $\Gamma \models_i t$ for all expressions t belonging to set s, then $\Gamma \models_i e$.

Definition 6 builds upon the above verification process. Below, r is a requirement, i.e., r $\in$ R. We represent each such requirement as a CTL formula.

Definition 6. *An nMAS $\Sigma = \langle L, A, M, N \rangle$, which generates a CTL model $\Gamma = \langle S, s_0, Act, \delta, Lbl \rangle$, is* correct *with respect to requirement r if and only if*

(1) A is consistent in Γ (all CTL formulas regarding assumptions hold);
(2) $\exists s_j \in S$ such that $\forall n_i \in N : sat_{n_i}$ is true in s_j (norms are consistent, that is, all norms can be satisfied);
(3) $\forall n_i \in N$: CTL formula ($\mathrm{AG}\ sat_{n_i} \longrightarrow \mathrm{AF}\ r$) is true in Γ.

Figure 6 visualizes Definition 6 using an example nMAS with one commitment and one prohibition. The enactment of nMAS starts in state s_0, where both the commitment and the prohibition are in their conditional states. According to Definition 6, we are interested only in states where all norms are satisfied (Condition 3). There must be at least one such state (Condition 2). All other paths are discarded (e.g., states with dashed circles that emanate from s_i). Satisfaction of individual norms is described in Section 3. Whenever the path reaches a state where all norms are satisfied (s_j or s_k), we check whether the requirement is satisfied in all paths that emanate from such a state. If the requirement is not satisfied in some path (the middle path emanating from state s_j with a double dashed circle), then the nMAS is incorrect. If no such state exists, and there exists a state where all norms are satisfied and the requirement is satisfied in all paths that emanate from that state (state s_k with double circle), then the nMAS is correct. We assume for the ease of drawing that the figure includes only those states where the assumptions hold (Condition 1).

3 NORM-BASED SPECIFICATION

An nMAS specification in Desen respects the grammar given in Table 1. Listing 1 shows such a specification for controlling access to a patient's EHR. Here, physician and hospital are agents, and assigned, EHR (meaning EHR is accessed), and logged_in are atomic propositions. physician is

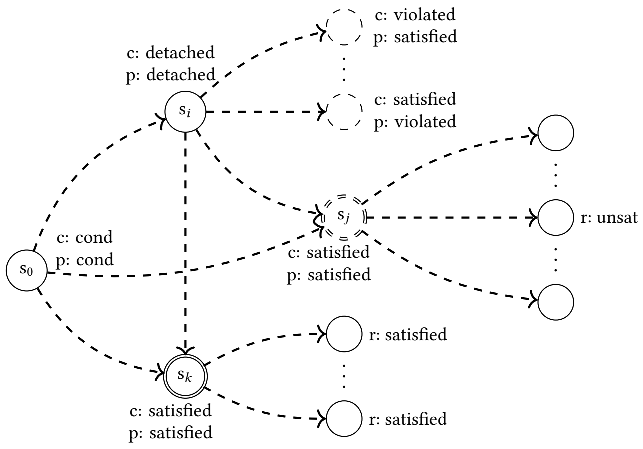


Fig. 6. Visualization of nMAS correctness. States shown as dashed circles are discarded from verification as they contain violated norms. Existence of a state with a double dashed circle implies incorrect nMAS as they lead to an unsatisfied requirement. A correct nMAS specification should consist of states shown as double circles as they do not lead to any unsatisfied requirements.

authorized to access the EHR if she is assigned to treat the patient (Line 1). PHYSICIAN needs to log in to the computer to access the EHR (Line 2).

Listing 1. A specification for PHYSICIAN activity.

```
1 a(PHYSICIAN, HOSPITAL, assigned, access_EHR)
2 m(logged_in, {access_EHR}, {})
```

We adopt model checking [Clarke et al. 1999] to formally verify whether an nMAS specification satisfies requirements (RQ_1). Algorithm 1 translates an nMAS specification (comprising roles, assumptions, mechanisms, and norms) into a NuSMV module comprising state transitions that follow mechanism and norm specifications.

The NuSMV syntax specifies a case statement for each variable (proposition) 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, we place transitions of the form "TRUE: X" last.

Algorithm 1 generates one case statement for each proposition $\phi \in \Phi$. 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 m's enabling condition is true and u cannot occur when m's enabling condition is true (Line 2). Similarly, we insert a transition for each v in m's delete list so as to prevent v from occurring when m's enabling condition is true (Line 3).

For each commitment c_i, we add a proposition (Line 5): sat_c_i to denote that c_i is satisfied. Note that such propositions are used in Definition 6 to verify the correctness of an nMAS. The transition rule for this proposition is described as follows: sat_c_i is true if Expr_2 is true (Line 6). This specification of a satisfied commitment is compatible with the commitment lifecycle of Figure 3. That is, a commitment is satisfied when its consequent is brought about.

For each prohibition p_j, we add a proposition (Line 8): sat_p_j to denote that p_j is satisfied. The transition rule for this proposition is described as follows: sat_p_j is true if $\overline{\text{Expr}_2}$ is true (Line 9).

Algorithm 1: T ⟵ generate(M, N)

```
   Input: M: mechanisms
   Input: N: norms
   Output: T: NuSMV transition model
1  foreach $m(enabler, add, delete) \in M$ do
2      Add transition “enabler: {TRUE, FALSE}; !enabler: FALSE;” for each proposition u ∈ add;
3      Add transition “enabler: FALSE;” for each proposition v ∈ delete;
4  foreach $c_i = c(\mathbb{R}_1, \mathbb{R}_2, Expr_1, Expr_2) \in N$ do
5      Add proposition sat_$c_i$;
6      Add transition “$Expr_2$: TRUE;” for sat_$c_i$;
7  foreach $p_j = (\mathbb{R}_1, \mathbb{R}_2, Expr_1, Expr_2) \in N$ do
8      Add proposition sat_$p_j$;
9      Add transition “$\overline{Expr_2}$: TRUE;” for sat_$p_j$;
10 foreach $a_k = a(\mathbb{R}_1, \mathbb{R}_2, Expr_1, Expr_2) \in N$ do
11     Add proposition sat_$a_k$;
12     Add transition “$Expr_2$: TRUE;” for sat_$a_k$;
13 T ⟵ “ ”; Group all transitions for $\phi$ into [transitions($\phi$)]; Add “next($\phi$):= case [transitions($\phi$)]
     esac;” to T;
14 return T;
```

That is, according to Figure 4, a prohibition is satisfied when its consequent is never brought about (i.e., $\overline{\text{Expr}_2}$ is true). Recall that we represent $\overline{\text{Expr}_2}$ as a temporal expression in CTL. Thus, we cannot directly implement that expression in NuSMV code. Instead, we assume the existence of distinct propositions that correspond to the nonoccurrence of events, and use those in the NuSMV implementation. For example, a proposition non_share_PHI represents the complement of share_PHI.

For each authorization a_k, we add a proposition (Line 11): sat_a_k to denote that a_k is satisfied. The transition rule for this proposition (Line 12) is similar to that for a commitment. According to the authorization lifecycle of Figure 5, an authorization is satisfied when its consequent is brought about.

Finally, we group transitions for each proposition and place them within a single case statement (Line 13).

Listing 2. NuSMV transitions generated for a commitment.

```
1 --c(PHY, HOS, emergency, treated ∧ operated)
2 next(sat_c):=
3  case
4   treated & operated: TRUE;
5  esac;
```

Listing 2 shows the resulting NuSMV specification for a commitment. In an emergency, PHYSICIAN is committed to treating and operating upon patients (Line 2). This commitment is satisfied when treated and operated are both true (Lines 3–6). Specifications for other norms are given similarly.

The formalization we have presented can be transformed into other formalizations to enable better adoption for practitioners who have no expertise in formal logic. However, please note that this is beyond the scope of this paper. Chopra and Singh [Chopra and Singh 2016a] propose a language to describe events and their effects on norms. We show the representation of a similar domain specific language (DSL) in Appendix B. Transformation of CTL formulas to more *practitioner-friendly* specifications have also been proposed. In particular, *specification patterns* [Attie et al. 1993; Dwyer et al. 1998] enable practitioners who have no expertise in formal logic to specify requirements and other formal system properties without the complications of CTL. An automated tool, ProPaS [Filipovikj et al. 2018], supports such specification for practitioners.

4 REFINEMENT VIA NORMATIVE DESIGN PATTERNS

More than merely verifying that an nMAS satisfies any stated requirements, we would like to specify an nMAS to ensure it satisfies the (possibly changing) stakeholder requirements (RQ_2). Accordingly, we propose a refinement process based on normative patterns. That is, if an nMAS is not correct with respect to some requirements, then our refinement process suggests refined sets of mechanisms and norms.

4.1 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 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 [Chopra and Singh 2009] propose a relation to compare commitments such that one is stronger than the other. We extend their definition to all norm types. Having a formal means of comparing norms enables us to refine them. Our notion of "refinement" is inspired by the alteration of formal method specifications via weakening or strengthening of preconditions and postconditions [Darimont and Van Lamsweerde 1996], and such refinement is supported by our patterns.

DEFINITION 7. *c_i = c(SBJ, OBJ, r, u) is stronger than c_j = c(SBJ, OBJ, s, v), denoted $c_i \gg c_j$, if and only if $s \vdash r$ and $u \vdash v$.*

A commitment becomes stronger if its antecedent becomes weaker (or stays the same) and its consequent becomes stronger (or stays the same). That is, a stronger commitment requires greater effort to fulfill the consequent. Consider c_1 = c(PHYSICIAN, HOSPITAL, true, operation ∧ clinic) and c_2 = c(PHYSICIAN, HOSPITAL, emergency, operation). Then, c_1 is stronger than c_2 because emergency ⊢ true and operation ∧ clinic ⊢ 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. *p_i = p(SBJ, OBJ, r, u) is stronger than p_j = p(SBJ, OBJ, s, v), denoted $p_i \gg p_j$, if and only if $s \vdash r$ and $v \vdash u$.*

A prohibition becomes stronger if both its antecedent and consequent become weaker (or stay the same). Consider p_1 = p(PHYSICIAN, HOSPITAL, true, share_PHI_family ∨ share_PHI_thirdparty) and p_2 = p(PHYSICIAN, HOSPITAL, emergency, share_PHI_thirdparty). Then, p_1 is stronger than p_2 because share_PHI_thirdparty ⊢ share_PHI_family ∨ share_PHI_thirdparty and emergency ⊢ true.

DEFINITION 9. *a_i = a(SBJ, OBJ, r, u) is stronger than a_j = a(SBJ, OBJ, s, v), denoted $a_i \gg a_j$, if and only if $s \vdash r$ and $v \vdash u$.*

An authorization becomes stronger if both its antecedent and consequent become weaker (or stay the same). That is, a stronger authorization allows improved flexibility. Consider a_1 = a(PHYSICIAN, HOSPITAL, true, own_patients ∨ other_patients) and a_2 = a(PHYSICIAN, HOSPITAL, true, own_patients). Then, a_1 is stronger than a_2 because their antecedents are identical and own_patients ⊢ own_patients ∨

other_patients. Note that a more restrictive authorization may provide "stronger" security. However, we use the word "stronger" based on the deontic semantics. Therefore, we define a stronger authorization to be the more flexible one. This definition enables our patterns to be applied in general settings.

Note that every norm is stronger than itself as $\vdash$ is reflexive. Moreover, if a stronger norm is satisfied in a CTL model, then all weaker norms are satisfied in that CTL model as well.

4.2 Patterns

Patterns provide ways of extending a given norm specification with specific conditions. The use of design patterns is common in software engineering [Zhu and Bayley 2013], and patterns has been investigated before in commitment-based specifications [Singh et al. 2009]. The strengthening and weakening patterns are inspired by healthcare law [Hammaker 2010], and focus on modifying antecedents and consequents of norms to either increase agents' flexibility, thereby promoting collaboration, or restricting agents to demote undesired outcomes. Table 2 shows an overview of our patterns.

Social patterns

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

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

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

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

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

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

Strengthening and weakening patterns expand available functionality, but can introduce vulnerabilities. Inspired by the sociolegal aspects of healthcare [Ham et al. 1988], the following *amendment* patterns address security and privacy concerns while promoting collaboration.

Responsibility replaces an authorization a_i = a(SBJ, OBJ, r, u) with a_j = a(SBJ, OBJ, s, v), where a_j is stronger than a_i, and adds a commitment $c_k(\text{SBJ}_k, \text{OBJ}_k, v, w)$. 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.

Consider the following scenario regarding the responsibility pattern: PHYSICIAN 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 HOSPITAL authorizes PHYSICIAN to use her phone to access EHR. However, doing so creates a vulnerability if PHYSICIAN leaves her phone unattended and forgets to lock it. Therefore, she has to commit to locking her phone to be granted this additional authorization. PHYSICIAN is accountable for a security breach she causes by not locking her phone.

Limitation replaces a prohibition p_i = p(SBJ, OBJ, r, u) with p_j = p(SBJ, OBJ, s, v), where p_i is stronger than p_j, and adds a prohibition $p_k(\text{SBJ}_k, \text{OBJ}_k, v, w)$. That is, Limitation both relaxes a prohibition and 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.

Note that the subjects (SBJ_k) and objects (OBJ_k) of the additional norms for the amendment patterns are not necessarily the same as the original norms. For example, allowing a physician to share a patient's PHI with colleagues increases the risk of PHI being disclosed. An additional

Table 2. Refinement patterns.

Pattern	Original	Refined	Conditions
Strengthening			
Liability	c_i = c(sbj, obj, r, u)	c_j = c(sbj, obj, s, v)	r ⊢ s, v ⊢ u
Grant	a_i = a(sbj, obj, r, u)	a_j = a(sbj, obj, s, v)	r ⊢ s, u ⊢ v
Denial	p_i = p(sbj, obj, r, u)	p_j = p(sbj, obj, s, v)	r ⊢ s, u ⊢ v
Weakening			
Release of Liability	c_i = c(sbj, obj, r, u)	c_j = c(sbj, obj, s, v)	s ⊢ r, u ⊢ v, $c_i \in \max(C)$
Revoke	a_i = a(sbj, obj, r, u)	a_j = a(sbj, obj, s, v)	s ⊢ r, v ⊢ u, $a_i \in \max(A)$
Accessibility	p_i = p(sbj, obj, r, u)	p_j = p(sbj, obj, s, v)	s ⊢ r, v ⊢ u, $p_i \in \max(P)$
Amendment			
Responsibility	a_i = a(sbj, obj, r, u)	a_j = a(sbj, obj, s, v)	r ⊢ s, u ⊢ v
		c_k = c(sbj_k, obj_k, v, w)	
		⟨trustworthy(sbj_k), true⟩	
Limitation	p_i = p(sbj, obj, r, u)	p_j = p(sbj, obj, s, v)	s ⊢ r, v ⊢ u, $p_i \in \max(P)$
		p_k = p(sbj_k, obj_k, v, w)	
		⟨trustworthy(sbj_k), true⟩	
Spawn			
Spawn[Commitment]		c(sbj, obj, r, u)	
Spawn[Authorization]		a(sbj, obj, r, u)	
Spawn[Prohibition]		p(sbj, obj, r, u)	
Overseer			
Overseer[Succeed]	c(sbj, obj, r, u)	c(sbj, obj, r, u)	
		c(sbj_k, obj, r, u)	sbj ≠ sbj_k
Overseer[NotSucceed]	p(sbj, obj, r, u)	p(sbj, obj, r, u)	
		c(sbj_k, obj, r, ¬u)	sbj ≠ sbj_k
Operational			
Enabler	$m_i = m$(r, u, v)	$m_j = m$(s, u, v)	r ⊢ s
Blocker	$m_i = m$(r, u, v)	$m_j = m$(s, u, v)	s ⊢ r
Spawn[Mechanism]		m(r, u, v)	
Sociotechnical			
Normify	$m_i = m$(r, {}, u)	$m_j = m$(s, {}, u)	r ⊢ s
		c(sbj, obj, v, w)	w ⊢ ¬u
		⟨trustworthy(sbj), true⟩	
Mechanize	c(sbj, obj, r, u)	m(r, u, {})	
		p(sbj, obj, r, w)	w ⊢ ¬u
	p(sbj, obj, r, u)	m(r, {}, u)	
		p(sbj, obj, r, w)	w ⊢ u

norm prohibits the physician's colleague (a new subject) from publishing the PHI online under the assumption that the colleague is trustworthy for not violating the prohibition.

Spawn[Norm] creates a norm. This pattern is especially useful for specifying an nMAS from scratch [Chopra et al. 2014; Günay et al. 2015] when the current set of norms cannot be strengthen or weakened to satisfy a stated requirement.

Overseer assigns a monitor to a given norm to ensure the norm is eventually satisfied. Given a commitment, Overseer[Succeed] specifies an additional commitment with a new subject to ensure that the consequent of the original commitment is brought about. Given a prohibition, Overseer[NotSucceed] specifies an additional commitment with a new subject to ensure that the consequent of the original prohibition is not brought about.

Technical patterns

Similar to the patterns applied on norms, the following *operational* patterns modify technical mechanisms.

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

The Enabler pattern refines a mechanism to relax an existing enabling condition. Consider m_1 = m(consent, {non_patient_EHR}, { }). The hospital's software allows PHYSICIAN to access the EHR of a patient the physician is not treating only if the patient has consented. We can refine this mechanism into m_2 = m(consent ∨ emergency, {non_patient_EHR}, { }), so that the software allows PHYSICIAN to access any patient's EHR during emergencies, instead of only those who consented.

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

The Blocker pattern refines a mechanism to make an existing enabling condition more strict.

Spawn[Mechanism] creates a mechanism.

Sociotechnical patterns

The following patterns are unique to Desen in that they enable refinements across the technical and social tiers by modifying mechanisms and norms at the same time, which other approaches cannot capture. We can thereby provide nMAS designers alternative means to satisfy stakeholder requirements by relying purely on regimentation via technical mechanisms, regulation via social norms, or a balanced combination of both. The following patterns are inspired by real data breaches in healthcare [HHS 2019; Kafalı et al. 2017b] and the means to mitigate them [Koppel et al. 2015].

Normify replaces m(r, { }, u) with m(s, { }, u) where r ⊢ s, and adds a commitment c(SBJ, OBJ, v, w) where w ⊢ ¬u. That is, a mechanism that puts a restriction on some operation is relaxed in the technical tier, and a norm is added in the social tier to ensure the operation is not performed in unintended situations. Consider mechanism m(inactive_15, { }, {logged_in}), which implements a session timeout on computers (physicians are automatically logged out after 15 minutes of inactivity). Now, this mechanism is refined into m(inactive_120, { }, {logged_in}), where the timeout duration is increased to two hours. Since this would create a security risk if the the computer is left unattended, a commitment c(PHYSICIAN, HOSPITAL, unattended, ¬logged_in) is added along with an assumption that the physician is trustworthy to satisfy the commitment.

The use of shared computers is common in healthcare practice, especially in medical emergencies. When technical controls are too restrictive, employees tend to find workarounds [Koppel et al. 2015] to evade such controls such as sharing passwords or writing them down. Those workarounds might lead to practices that are noncompliant with the underlying healthcare security and privacy laws. Therefore, such practices should be avoided. The normify pattern implements a transition from the technical tier to the social tier to reduce the need for such workarounds while maintaining accountability via norms. That is, normify provides a relaxation in the technical controls while providing additional social regulation, which enables employees flexible execution of their everyday tasks.

Mechanize replaces commitment c(SBJ, OBJ, r, u) with m(r, u, { }) or prohibition p(SBJ, OBJ, r, u) with m(r, { }, u), and adds a prohibition p(SBJ, OBJ, r, w) where w ⊢ u. That is, a commitment or a prohibition in the social tier that requires a manual operation to be performed is removed, and a mechanism is added in the technical tier to automate that operation. Moreover, a prohibition is added to ensure the new mechanism is not tampered with. Consider commitment c(PHYSICIAN, HOSPITAL, public_computer, clear_cache), which means that physicians are committed to clearing the cache after using a public computer. Now, this commitment is refined into mechanism *m*(public_computer, {clear_cache}, {}) which implements a plugin to automatically clear the cache. In addition, a prohibition p(PHYSICIAN, HOSPITAL, public_computer, remove_plugin) is added to ensure physicians do not remove the plugin from a public computer.

Let us consider another example of a sociotechnical pattern. In order to ensure that computers are not left unattended, a new plugin is installed on the physicians' computers to monitor their activities. This new plugin, however, could result in the physicians' privacy being violated, and hence physicians are now authorized to disable the plugin for any personal use to prevent leakage of sensitive data.

The use of shared electronic devices (e.g., computers and printers) is common practice in various industrial settings and can cause data breaches due to human error. For example, a simple failure to erase a photocopier's cache led to the disclosure of over 300,000 patient records in 2010 [HHS 2019]. The mechanize pattern implements a transition from the social tier to the technical tier by automating a previously manual practice to reduce such human errors, e.g., by installing a plugin to periodically clear the cache.

4.3 Generating Refined Specifications

In practice, STS designers can use patterns for creating and revising nMAS specifications. Our aim is to provide generic templates to guide the design process. Our templates can be customized by designers to address requirements that demand finer regulation and control. For example, the additional commitment in normify can instead be replaced with a prohibition. Following the session timeout example, the hospital might prohibit physicians from leaving a computer unattended if the physician is still logged in to the computer.

We now formulate the problem of generating refined specifications in abstract terms. For this purpose, we adopt the notion of classical AI planning [Russell and Norvig 1995]. The initial state s_0 is described by a tuple $\langle A, M, N\rangle$. Suppose $A, M, N \nvdash R$. That is, the initial assumptions, mechanisms, and norms do not satisfy all requirements. A goal state s_g is described by a tuple $\langle A', M', N'\rangle$, where $A', M', N' \vdash R$. That is, a goal state satisfies all requirements. Note that there might be multiple states that satisfy all requirements. S_G is the set of all such states.

A solution to the problem of generating refined specifications is a sequence of pattern operations that takes s_0 to an s_g. In essence, each solution is a plan. Note that not every solution is equally good, and it is helpful to find refinements that make minimal changes to a prior specification. Selecting among possible solutions would generally involve domain-specific considerations, potentially captured as heuristics. One such heuristic is to prefer the solution with the least number of pattern operations (i.e., shortest path). Another heuristic would be based on a distance function, $\Delta_{s_g - s_0}$, to compute the distance of a goal state from the initial state. We present below two such heuristics for guiding the STS refinement process. Such heuristics are compatible with studies suggesting that higher costs associated with implementation of drastic changes [Suri 2011] and substantial reorganization requirements [Hall 2004] negatively impact the rate of technology adoption. Therefore, a realistic planning solution would prioritize revisions with as few patterns as required or patterns with low implementation costs.

Following a heuristic-based approach, our formulation of the refinement problem enables the potential application of more sophisticated planning techniques, which however are outside of our present scope. Let us consider the following heuristics for computing the *cost* and *risk* associated with a given refinement solution.

HEURISTIC 1. Cost: The cost of implementing a mechanism is higher than the cost of enforcing a norm.

HEURISTIC 2. Risk: The risk associated with the potential violation of a norm is higher than the potential malfunctioning of a mechanism.

Note that the above heuristics do not take into account the predicates involved in the mechanisms or norms. A domain ontology [Kafalı et al. 2017b] can be used to further associate costs and risk with individual predicates. We describe how Heuristics 1 and 2 are applied in an emergency healthcare scenario in Section 5.

Next, we demonstrate how a pattern can be applied as an operation. Let $s_i = \langle A_i, M_i, N_i \rangle$ be an intermediate state, and authorization $a_i = a(\mathbb{R}_1,\mathbb{R}_2, t \wedge z, u) \in N_i$. Then, the Grant pattern can be applied as an operation to s_i by adding a proposition to the antecedent or consequent of a_i or removing a proposition from the antecedent or consequent of a_i using the set of domain propositions. The alternative successor states of s_i would include an authorization from the following set: $\{a(\mathbb{R}_1,\mathbb{R}_2, t, u)\} \cup \{a(\mathbb{R}_1,\mathbb{R}_2, z, u)\} \cup \{a(\mathbb{R}_1,\mathbb{R}_2, (t \wedge z) \vee w, u) | \ w \in \Phi \setminus \{t, z, u\}\} \cup \{a(\mathbb{R}_1,\mathbb{R}_2, t \wedge z, u \vee v) | \ v \in \Phi \setminus \{t, z, u\}\}$. Note that the set of domain propositions used for applying the pattern can be further restricted to only the propositions relevant to the given requirement. Doing so would significantly reduce the space of successor states. After each step, a solver verifies (e.g., via a model checker) whether the current state satisfies all requirements (i.e., a goal state is reached). If so, the plan is added to the set of solutions.

5 EVALUATION: HEALTHCARE PRIVACY SCENARIO

Recall the five methodological steps described in the "Practical usage and implications" section. This realistic use case enables us to validate our model checking tool (Step i) and refinement patterns (Step iv). To demonstrate how refinement works, we formalize the scenario described in Example 1 using the assumptions, mechanisms, and norms listed in Table 3 (see Appendix A for the complete NuSMV implementation).

Assumptions: s_1 means that it is not possible to obtain consent from patients in national emergencies; s_2 means that it is not declared a national emergency in a hospital unless the hospital is located in the emergency area and an emergency is currently active; s_3 means that the subject of a commitment brings about the commitment's consequent if the commitment is detached and the subject is trustworthy; s_4 means that the subject of a prohibition does not bring about the prohibition's consequent if the prohibition is detached and the subject is trustworthy; s_5 and s_6, respectively, mean that physicians and outside physicians are trustworthy.

Mechanisms: m_1 means that a national emergency is declared when the President and the Secretary of Health both declare an emergency situation; m_2 means that a patient's consent to share PHI is overridden when the patient requests confidential communications during regular medical practice (nonemergency); m_3 means that a patient can give consent at any time.

Norms: c_1 means that a physician is committed to a patient to accommodating the patient's request if the patient requests confidential communications and it is not a national emergency; a_1 means that a patient is authorized by the hospital to request confidential communications when it is not a national emergency; p_1 means that a physician is prohibited by the hospital from sharing a patient's PHI with third parties if the patient does not give consent; p_2 means that a physician is prohibited by the hospital from sharing a patient's PHI with an outside physician if the patient does not give consent and it is not a national emergency; p_3 means that a physician is prohibited by the hospital from sharing a patient's PHI with the patient's family if the patient does not give consent and it is not a national emergency; p_4 means that an outside physician is prohibited by the hospital from sharing a patient's PHI with third parties at all times (unconditional norm).

Table 3. Specification of Example 1 in DESEN.

	Assumptions
s_1	$\langle \neg$consent, emergency$\rangle$
s_2	$\langle \neg$emergency, $\neg$emergency_area $\vee$ $\neg$emergency_period$\rangle$
s_3	$\langle$con, c(SBJ, OBJ, ant, con) $\wedge$ ant $\wedge$ trustworthy(SBJ)$\rangle$
s_4	$\langle \neg$con, p(SBJ, OBJ, ant, con) $\wedge$ ant $\wedge$ trustworthy(SBJ)$\rangle$
s_5	$\langle$trustworthy(PHYSICIAN), true$\rangle$
s_6	$\langle$trustworthy(OUTSIDE_PHYSICIAN), true$\rangle$
	Mechanisms
m_1	*m*(president_declare $\wedge$ secretary_HHS_declare, {emergency}, {})
m_2	*m*(confidential_communication $\wedge$ $\neg$emergency, {}, {consent})
m_3	*m*(true, {consent}, {})
	Norms
c_1	c(PHYSICIAN, PATIENT, confidential_communication $\wedge$ $\neg$emergency, accommodate_request)
a_1	a(PATIENT, HOSPITAL, $\neg$emergency, confidential_communication)
p_1	p(PHYSICIAN, HOSPITAL, $\neg$consent, share_PHI_thirdparty)
p_2	p(PHYSICIAN, HOSPITAL, $\neg$consent $\wedge$ $\neg$emergency, share_PHI_outside_physician)
p_3	p(PHYSICIAN, HOSPITAL, $\neg$consent $\wedge$ $\neg$emergency, share_PHI_family)
p_4	p(OUTSIDE_PHYSICIAN, HOSPITAL, true, share_PHI_thirdparty)

Requirements: We consider the following HIPAA requirements [HHS 2014] that are relevant to Example 1.

R-Disclose: A patient's PHI must not be shared with third parties without the patient's consent. In CTL:

AG $\neg$($\neg$consent $\wedge$ share_PHI_thirdparty)

There cannot be a time point on any path where there is no consent from the patient but PHI is shared with third parties.

R-Share: In case of a national emergency, physicians must be allowed to share a patient's PHI with the patient's family members. This reflects the waiving of the HIPAA clause 45 CFR 164.510(b) [HHS 2014] in emergencies: "the requirements to obtain a patient's agreement to speak with family members or friends involved in the patient's care." In CTL:

EF ((emergency $\wedge$ $\neg$consent) $\wedge$ share_PHI_family)

There must be at least one path where a patient's PHI is shared, when there is a national emergency but no consent from the patient.

R-Communication: During regular medical practice, patients must be allowed to request confidential communications regarding their PHI. This reflects the HIPAA clause 45 CFR 164.522(b) [HHS 2014]:

"the patient's right to request confidential communications." Two CTL formulas together represent this requirement:

EF (¬emergency ∧ confidential_communication)
There must be at least one path where a patient can request confidential communications in case of a nonemergency.

AG (emergency ⟶ AF ¬confidential_communication)
From any time point when there is a national emergency, there must not be any confidential communications at any path.

Note that we do not tackle requirements elicitation in this paper. To demonstrate how Desen refines specifications, we begin from an nMAS specification that does not satisfy some of the requirements. By applying our patterns, we end with the specification in Table 3.

Let us begin by describing the initial specification. We use subscript i (short for initial) to distinguish from the correct specification in Table 3.

m_1 = m(president_declare ∧ secretary_HHS_declare, {emergency}, {})

m_{2i} = m(confidential_communication, {}, {consent})

m_3 = m(true, {consent}, {})

c_1 = c(PHYSICIAN, PATIENT, confidential_communication ∧ ¬emergency, accommodate_request)

a_{1i} = a(PATIENT, HOSPITAL, true, confidential_communication)

p_{1i} = p(PHYSICIAN, HOSPITAL, ¬consent, share_PHI_thirdparty ∨ share_PHI_outside_physician ∨ share_PHI_family)

Since disclosing the patient's PHI is prohibited via p_{1i}, requirement R-Disclose is satisfied. Moreover, this specification does not allow any flexibility in national emergency situations. Thus, R-Share is not satisfied. Moreover, mechanism m_{2i} and authorization a_{1i} are given in broad terms and do not capture communications in national emergencies, which leaves R-Communication unsatisfied (the first CTL formula for R-Communication is true, whereas the second CTL formula for R-Communication is false).

Below, we present the refinement steps (i.e., pattern operations) of two potential solutions among all possible candidate solutions. Solution 1 is norm-based, which yields lower cost (due to Heuristic 1), but higher risk (due to Heuristic 2). Solution 2 is mechanism-based, which yields higher cost, but lower risk (again based on the heuristics). Figure 7 depicts the pattern operations involved in the solutions.

Solution 1. Norm-based:

(1) Accessibility: Prohibition p_{1i} is refined into p_1, p_2, and p_3 to improve physicians' flexibility during emergencies by enabling PHYSICIAN to share a patient's PHI with OUTSIDE_PHYSICIAN or the patient's family without consent.
(2) Limitation: Note that the above refinement satisfies R-Share, but violates R-Disclose because outside physicians may disclose a patient's PHI. Therefore, an additional prohibition p_4 is specified to make outside physicians accountable for such disclosures.
(3) Revoke: NuSMV verifies that the second CTL formula for R-Communication is false. Therefore, a_{1i} is refined into a_1 using the Revoke pattern.

Solution 2. Mechanism-based:

(1) Mechanize: Prohibition p_{1i} is removed, and a mechanism m(¬consent, {}, {share_PHI_thirdparty, share_PHI_outside_physician, share_PHI_family}) is added. By replacing the norm with a mechanism, physicians are prevented from downloading or copying the content on their screen for sharing purposes, e.g., via a plugin installed on their computer.
(2) Limitation: An additional prohibition p_4 is specified to make outside physicians accountable for disclosures in case the mechanism on their computer fails to prevent them from sharing patient information.
(3) Blocker: Regarding requirement R-Communication, m_{2i} is refined into m_2 using the Blocker pattern.

6 EVALUATION: MODELER STUDY

We conducted a human-subject study to evaluate the effectiveness of the refinement patterns in specifying an nMAS. Our study was declared exempt by NC State University's Institutional Review Board. We circulated an advertisement for participation, and asked interested participants to complete a pre-participation survey. Based on their responses, we selected 32 graduate computer science students as study participants, and created two groups (*Control* and *DESEN*) 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 (see Appendices C–G for study details). Table 4 summarizes the demographics information of our study participants that we collected in the pre-participation survey.

Table 4. Demographics information of the study participants.

Education (degree pursuing)	MS: 93.33%, PhD: 6.67%
Academic (programming and software development) experience	Less than 3 months: 0%, 3 months–1 year: 10%, 1–3 years: 26.66%, More than 3 years: 63.33%
Industry work experience	Less than 3 months: 33.33%, 3 months–1 year: 23.33%, 1–3 years: 26.66%, More than 3 years: 16.66%
Familiarity with conceptual modeling	Not familiar: 10%, Familiar with concepts, but no practical experience: 43.33%, Familiar and some academic experience: 20%, Familiar and some industry experience: 26.67%
Familiarity with norms	Not familiar: 40%, Familiar with concepts, but no practical experience: 33.33%, Familiar and some academic experience: 13.33%, Familiar and some industry experience: 13.33%

6.1 Study Design

We follow a one-factor two-alternatives design. Participants in the DESEN group are guided with refinement patterns (listed in Appendix D.2), whereas participants in the Control group are given a basic definition of refinement (listed in Appendix D.1). Our study has three phases.

Learn. Participants learn how to define requirements and specify an nMAS for a healthcare privacy scenario.

Create Participants define requirements for a healthcare security scenario and produce its nMAS specification.

Requirements: {R-Disclose, ~~R-Share~~, ~~R-Communication~~}
Assumptions:
{s_1 = ⟨¬consent, emergency⟩, s_2 = ⟨¬emergency, ¬emergency_area ∨ ¬emergency_period⟩,
s_3 = ⟨con, c(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩, s_4 = ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩,
s_5 = ⟨trustworthy(PHYSICIAN), true⟩}
Mechanisms:
{m_1 = m(president_declare ∧ secretary_HHS_declare, {emergency}, {}), m_{2i} = m(confidential_communication, {}, {consent}),
m_3 = m(true, {consent}, {})}
Norms:
{c_1 = c(PHYSICIAN, PATIENT, confidential_communication ∧ ¬emergency, accommodate_request),
a_{1i} = a(PATIENT, HOSPITAL, true, confidential_communication),
p_{1i} = p(PHYSICIAN, HOSPITAL, ¬consent, share_PHI_thirdparty ∨ share_PHI_outside_physician ∨ share_PHI_family)}

↓ Accessibility and Mechanize patterns: p_{1i} removed, p_1, p_2, p_3, and m_4 added

Requirements: {~~R-Disclose~~, R-Share, ~~R-Communication~~}
Assumptions:
{s_1 = ⟨¬consent, emergency⟩, s_2 = ⟨¬emergency, ¬emergency_area ∨ ¬emergency_period⟩,
s_3 = ⟨con, c(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩, s_4 = ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩,
s_5 = ⟨trustworthy(PHYSICIAN), true⟩}
Mechanisms:
{m_1 = m(president_declare ∧ secretary_HHS_declare, {emergency}, {}), m_{2i} = m(confidential_communication, {}, {consent}),
m_3 = m(true, {consent}, {}), m_4 = m(¬consent, {}, {share_PHI_thirdparty, share_PHI_outside_physician, share_PHI_family})}
Norms:
{c_1 = c(PHYSICIAN, PATIENT, confidential_communication ∧ ¬emergency, accommodate_request),
a_{1i} = a(PATIENT, HOSPITAL, true, confidential_communication),
p_1 = p(PHYSICIAN, HOSPITAL, ¬consent, share_PHI_thirdparty),
p_2 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_outside_physician),
p_3 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_family)}

↓ Limitation pattern: p_4 and s_6 added

Requirements: {R-Disclose, R-Share, ~~R-Communication~~}
Assumptions:
{s_1 = ⟨¬consent, emergency⟩, s_2 = ⟨¬emergency, ¬emergency_area ∨ ¬emergency_period⟩,
s_3 = ⟨con, c(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩, s_4 = ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩,
s_5 = ⟨trustworthy(PHYSICIAN), true⟩, s_6 = ⟨trustworthy(OUTSIDE_PHYSICIAN), true⟩}
Mechanisms:
{m_1 = m(president_declare ∧ secretary_HHS_declare, {emergency}, {}), m_{2i} = m(confidential_communication, {}, {consent}),
m_3 = m(true, {consent}, {}), m_4 = m(¬consent, {}, {share_PHI_thirdparty, share_PHI_outside_physician, share_PHI_family})}
Norms:
{c_1 = c(PHYSICIAN, PATIENT, confidential_communication ∧ ¬emergency, accommodate_request),
a_{1i} = a(PATIENT, HOSPITAL, true, confidential_communication),
p_1 = p(PHYSICIAN, HOSPITAL, ¬consent, share_PHI_thirdparty),
p_2 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_outside_physician),
p_3 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_family),
p_4 = p(OUTSIDE_PHYSICIAN, HOSPITAL, true, share_PHI_thirdparty)}

↓ Revoke and Blocker patterns: m_{2i} and a_{1i} removed, m_2 and a_1 added

Requirements: {R-Disclose, R-Share, R-Communication}
Assumptions:
{s_1 = ⟨¬consent, emergency⟩, s_2 = ⟨¬emergency, ¬emergency_area ∨ ¬emergency_period⟩,
s_3 = ⟨con, c(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩, s_4 = ⟨¬con, p(SBJ, OBJ, ant, con) ∧ ant ∧ trustworthy(SBJ)⟩,
s_5 = ⟨trustworthy(PHYSICIAN), true⟩, s_6 = ⟨trustworthy(OUTSIDE_PHYSICIAN), true⟩}
Mechanisms:
{m_1 = m(president_declare ∧ secretary_HHS_declare, {emergency}, {}),
m_2 = m(confidential_communication ∧ ¬emergency, {}, {consent}), m_3 = m(true, {consent}, {}),
m_4 = m(¬consent, {}, {share_PHI_thirdparty, share_PHI_outside_physician, share_PHI_family})}
Norms:
{c_1 = c(PHYSICIAN, PATIENT, confidential_communication ∧ ¬emergency, accommodate_request),
a_1 = a(PATIENT, HOSPITAL, ¬emergency, confidential_communication),
p_1 = p(PHYSICIAN, HOSPITAL, ¬consent, share_PHI_thirdparty),
p_2 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_outside_physician),
p_3 = p(PHYSICIAN, HOSPITAL, ¬consent ∧ ¬emergency, share_PHI_family),
p_4 = p(OUTSIDE_PHYSICIAN, HOSPITAL, true, share_PHI_thirdparty)}

Fig. 7. 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.

Maintain. Participants comprehend and maintain requirements of an nMAS specification for a security access control scenario adopted from Tsigkanos et al. [2014].

Data Collection and Metrics: The study was carried out using pen and paper without tool support. Participants record completion times and complete a post-study survey regarding the helpfulness of patterns in their specification. The first two authors designed an oracle solution for each phase, and marked participants' specifications using the following metrics.

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

Correctness of specification: Fraction of participant-stated norms that occur in the oracle. Higher is better.

Time to define specification: Time in minutes recorded by participants. Lower is better.

Ease of defining specification: Subjective ratings provided by the participants via the post-study survey on a Likert scale ranging from very hard (1) to very easy (5). Higher is better.

Hypotheses: We propose four two-sided hypotheses to compare how the Desen and Control treatments influence the production of nMAS specifications. For each hypothesis, a null hypothesis (omitted for brevity) states that there is no difference between the groups.

H1 Desen and Control differently influence the coverage of specifications produced by study participants.

H2 Desen and Control differently influence the correctness of specifications produced by study participants.

H3 Desen and Control differently influence the time expended by study participants to produce specifications.

H4 Desen and Control differently influence participants' subjective ease of producing specifications.

Moreover, we propose subhypotheses to evaluate the influence of Desen on coverage, correctness, speed of producing specifications by participants with or without prior knowledge or experience of working with norms (Appendix C provides details).

6.2 Results

We analyze specifications produced by participants in the Control and Desen groups, and compute the means of coverage and correctness of the specifications, and the means of time to define specification and medians of ease of defining specification values reported by the participants. We perform a two-tailed t-test to test the significance of hypotheses H1, H2, H3, and Wilcoxon's rank-sum test for H4. We choose Wilcoxon's test for H4 as it has a power advantage over the t-test for Likert scale data. Specifically, Wilcoxon's test compares medians, whereas t-test compares means [Hollander and Wolfe 1999]. Following recent recommended practice, we provide descriptive statistics along with p-values. We adopt 5% as the significance cutoff for p-values, below which each null hypothesis is rejected. To measure the effect size for the difference between means, we compute Hedges' g, which is well-suited to measuring effect for small (and unequal) sample sizes [Grissom and Kim 2012].

Coverage and correctness of specification. We evaluate H1 and H2 by computing the coverage and correctness metrics of specifications produced by participants. The mean coverage (65%) and correctness (67%) for the specifications produced using Desen was higher (but not significant at 5%) than the mean coverage (62%) and correctness (62%) for those of Control. We further analyzed specifications to see if prior knowledge or experience with norms has an effect. We observe significant (at 5%) gains for participants with no prior industry experience in conceptual modeling and no knowledge of norms with large effect size [Grissom and Kim 2012]. We attribute to the fact that all participants were introduced to the basics of norm refinement during the study. Figures 8 and 9 show the box plots for coverage (left) and correctness (right) of specifications produced by the participants with low experience in conceptual modeling and no prior knowledge of norms.

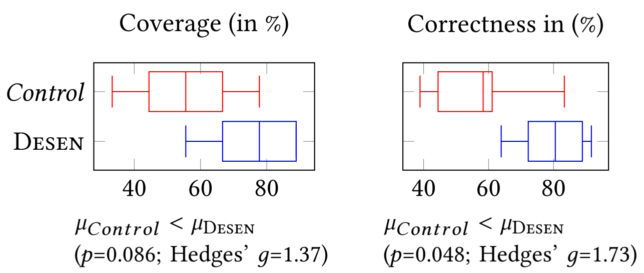


Fig. 8. Participants with low experience in conceptual modeling.

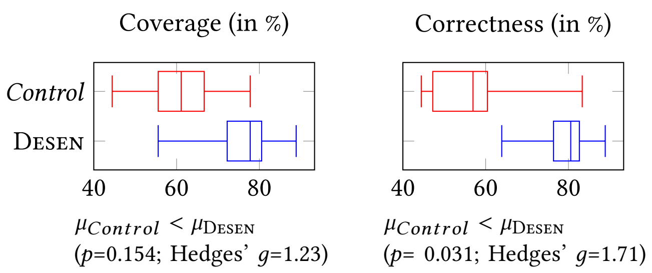


Fig. 9. Participants with no prior experience in norms.

Time to produce specification. We evaluate H3 by comparing the time expended by participants to produce specifications. The mean time to produce specifications using Desen (63 minutes) was higher (but not significant at 5%) than using Control (58 minutes). The result is not surprising as participants in the Desen group were required to first get acquainted with the normative design patterns. Further analysis suggests that, although Desen takes more time in the learning phase for participants with no prior knowledge of norms, they speed up in the defining specification and maintenance phases.

Ease of defining specification. We evaluate H4 based on the post-survey completed by participants after each phase wherein they answered questions on the ease of defining specifications either using Desen or Control. Whereas participants found both Desen and Control equally easy to work with, participants in the Desen group reported that patterns were helpful in their specification (μ=4.4/5), and stated that an automated tool to generate alternative refinements would improve accuracy (μ=4.27/5).

6.3 Practical Implications

Both study groups employed a normative approach to capture requirements, and achieved good coverage and accuracy results (excluding minor errors regarding the logical representation), which suggests that norms provide a promising way of capturing requirements. We did not employ an automated tool for Desen, because we wanted to evaluate the usefulness of patterns without coupling it with the effect of using a tool. We defer such a study to future work.

6.4 Study Threats and Mitigation

We identified and mitigated two threats. *(i) Skill differences:* We surveyed participants for their education and prior experience with conceptual modeling and software engineering industry experience. We balanced Control and Desen with respect to aggregate experience of participants based on this survey. *(ii) Failure to report information:* Participants completed the time and difficulty survey for a phase while it was fresh in their minds. *(iii) Internal validity of learning material:* Participants in the Control group and Desen group were provided with the same tutorial on normative requirements. Participants of the Desen group were additionally provided with a refinement patterns guide. *(iv) Construct validity of surveys: (v) External validity for using students instead of practitioners:* Since producing specification is a software engineering task performed by practitioners, our participants—a large portion of whom have industry work experience—are acceptable surrogates for software engineers.

7 EVALUATION: SIMULATION EXPERIMENT

Recall the five methodological steps described in the "Practical usage and implications" section. The simulation experiments enable us to evaluate the solutions gathered from our modeler study against benchmark solutions (Steps ii and iii). We adopt MASON [Luke et al. 2005], which is a Java-based multiagent simulation library, to develop a multiagent simulation environment based on an extension of the healthcare privacy scenario described in Section 5.

7.1 Community Healthcare Environment

The environment represents a community-based social network consisting of several communities, in each of which several residents (agents) reside. Each community has one community hospital, and each community hospital has several physicians (agents).

In the community healthcare environment, time is represented in steps. At each step there is a probability (*illness probability*) of a resident to fall ill. When a resident falls ill, the health of the resident (*resident health*) starts deteriorating. Illness in our environment is of two types—*critical* and *noncritical*. An ill resident residing in a community visits the community hospital for treatment.

An ill resident admitted to the community hospital is a patient (agent). Each community hospital has a capacity—the maximum number of patients it can admit at one time step—depending upon number of physicians and their availability. The hospital assigns an available physician to a patient. To start treatment, a physician accesses a patient's PHI. A physician can either treat one critical patient or three noncritical patients. After a physician starts treatment, a patient takes three time steps to recover from a critical illness, and one time step to recover from a noncritical illness. A critically ill patient may recover faster (*early-recovery probability*) in two time steps if the family is aware of the illness. When a patient's treatment is completed, the patient is discharged from the hospital.

Communities in the environment often suffer from a disease outbreak—at each step there is a probability (*outbreak probability*) of a disease outbreak. When such an outbreak occurs, all residents who fall ill suffer from a critical illness at the same time step. During and after the outbreak period, if the community's average health falls below a threshold (*low community health threshold*), the community declares an emergency, and the community hospital starts operating under emergency operating procedures. A community hospital operating under an emergency requests external physicians. When the community average health rises above a threshold (*high community health threshold*), the emergency status is withdrawn and the community hospital returns to operating under normal procedures and the external physicians are released.

Table 5 summarizes the simulation parameters and their values.

Table 5. Simulation parameters and their values or range.

Parameter		**Value**
Number of residents		100
Hospital		1
Number of internal physicians	low	15
	high	25
Number of external physicians	low	5
	high	15
Illness types		critical, noncritical
Illness probability	low	0.10
	high	0.25
Outbreak probability	low	0.10
	high	0.25
Early-recovery probability		0.25
Treatment time	critical	3
	critical (early recovery)	2
	noncritical	1
Internal physician compliance		0.90
External physician compliance	low	0.70
	high	0.90
Mechanism availability	low	0.70
	high	0.90
Resident health		0–100
Community health		0–100
Community health threshold	low	75
	high	85
Family PHI request probability		0.50

7.2 Requirements and Solutions

Hospitals are bound by law to keep their patient's electronic health records (EHR) private. Thus, internal and external physicians working in a community hospital must not publish a patient's protected health information (PHI) online. External physicians are prohibited to access a patient's PHI directly. However, internal physicians may share a patient's PHI with external physicians when the hospital is operating under emergency. Physicians may share a patient's PHI with the patient's family when there is an emergency [HHS 2014].

Requirements:

R-Publish: A patient's PHI should not be published online with personally identifying information under any circumstances.

R-External: In emergencies, hospital physicians may share a patient's PHI with outside physicians to cope with the load.

R-Family: In emergencies, hospital physicians may share patient's PHI with family members to inform family members or gather new information to help with treatment.

Initial STS:

(1) p_{1i} = p(PHYSICIAN, HOSPITAL, true, publish_PHI_online)
(2) p_{2i} = p(PHYSICIAN, HOSPITAL, true, share_PHI_outside_physician)
(3) a_{1i} = a(PHYSICIAN, HOSPITAL, true, share_PHI_family)

Based on the above initial STS specification, we first describe the benchmark solutions that we use to evaluate the modeler study solutions against.

Spec$_1$. NORM – Norm-based solution
(1) p_{1i} stays the same.
(2) Accessibility: Refine p_{2i} into p_2 = p(PHYSICIAN, HOSPITAL, ¬ emergency, share_PHI_outside_physician)
(3) Revoke: Refine a_{1i} into a_1 = a(PHYSICIAN, HOSPITAL, emergency, share_PHI_family)

Spec$_2$. MECH – Mechanism-based solution
(1) Mechanize: Replace p_{1i} with m_1 = m(true, {}, {publish_PHI_online})
(2) Mechanize: Replace p_{2i} with m_2 = m(emergency, {share_PHI_outside_physician}, {})
(3) Mechanize: Replace a_{1i} with m_3 = m(emergency, {share_PHI_family}, {})

Spec$_3$. STOS – Sociotechnical optimization solution
(1) Mechanize: Replace p_{1i} with m_1 = m(true, {}, {publish_PHI_online})
(2) Accessibility: Refine p_{2i} into p_2 = p(PHYSICIAN, HOSPITAL, ¬ emergency, share_PHI_outside_physician)
(3) Revoke: Refine a_{1i} into a_1 = a(PHYSICIAN, HOSPITAL, emergency, share_PHI_family)

Next, we describe the solutions provided by the participants in our modeler study. Our motivation behind this experiment is to evaluate the quality of the specifications produced by the modeler study using simulation (in addition to the correctness evaluation performed in Section 6). We apply a set of selection criteria to construct specifications from participants' solutions, simulate those solutions, and report the results using our metrics. We take Scenario A (Section F.1) as a basis for comparison as it constitutes the most relevant setting to simulation environment. More specifically, Scenario A shares the same set of requirements (R-publish, R-external, R-family) and a similar set of propositions (publish_PHI_online, share_PHI_outside_physician, share_phi_colleague, share_PHI_family) as the community healthcare setting. We discard Scenario B (Section F.2) and Scenario C (Section F.3) since they constitute different settings. However, we check participants' solutions in those scenarios to validate consistency. Below, we list the selection criteria:

C$_1$ – Primary. Discard participant solutions with more than one norm incorrectly specified in Scenario A.

C$_2$ – Secondary. Discard participant solutions with a majority of the requirements and norms incorrectly specified for Scenarios B and C.

C$_3$ – Secondary for DESEN. Discard participant solutions that do not involve pattern usage.

C$_4$ – Secondary for Control. Discard participant solutions that include redundant norms.

Following C$_1$ and C$_2$, we selected nine participant solutions for DESEN and five participant solutions for Control. Following C$_3$, we discarded five of the nine participant solutions for DESEN, thus ending up with four participant solutions. Following C$_4$, we discarded two of the five participant solutions for Control, thus ending up with three participant solutions. As a result, we construct one solution for each participant group reflecting the solutions from their individual participants.

Spec$_4$. Desen – Most general solution for the Desen participant group with an additional norm, Commitment (5), appearing in at least two solutions.

(1) Accessibility: p(PHYSICIAN, HOSPITAL, true, publish_PHI_online)
(2) Spawn: a(PHYSICIAN, HOSPITAL, true, share_phi_colleague)
(3) Revoke: a(PHYSICIAN, HOSPITAL, true, share_PHI_family)
(4) Spawn: p(PHYSICIAN, HOSPITAL, ¬emergency, share_PHI_family)
(5) Responsibility: c(PHYSICIAN, HOSPITAL, emergency, share_PHI_family)

Spec$_5$. Control Unanimous solution among all the Control group participants.

(1) p(PHYSICIAN, HOSPITAL, true, publish_PHI_online)
(2) a(PHYSICIAN, HOSPITAL, true, share_phi_colleague)
(3) a(PHYSICIAN, HOSPITAL, emergency, share_PHI_family)
(4) p(PHYSICIAN, HOSPITAL, ¬emergency, share_PHI_family)

Note that our modeler study instructions for the participants in the Desen group enabled some of them to specify an additional commitment using the "Responsibility" pattern. The resulting specification allows physician agents in our simulation to be proactive, i.e., to contact the patient's family to share PHI with a chance to enable early recovery for the patient, compared to the Control group solution, where agents remain reactive, i.e., must wait for a request from the patient's family to share PHI.

7.3 Metrics

To measure the effectiveness of the solutions, we compute the following metrics.

Information disclosure is the average number of unauthorized disclosures by physicians in a community hospital. Lower is better.

Life sustenance is the number of residents still alive in a community. Range: 0–100. Higher is better.

Social welfare is the average community health. Range: 0–100. Higher is better.

7.4 Hypotheses

We propose three null hypotheses that state there are no differences between the five solutions—Norm, Mech, Stos, Desen, and Control. The alternative hypotheses indicate there is a difference.

H5$_{\text{disclosure}}$. There is no difference in the number of unauthorized disclosures that result by the use of five solutions—three benchmarks, Desen, and Control solutions.

H6$_{\text{sustenance}}$. There is no difference in the life sustenance yielded by the five solutions—three benchmarks, Desen, and Control solutions.

H7$_{\text{welfare}}$. There is no difference in the social welfare yielded by the five solutions—three benchmarks, Desen, and Control solutions.

7.5 Experiments and Results

We evaluate the hypotheses via multiple experiments on the community healthcare environment in which we simulate each of the five solutions (Norm, Mech, Stos, Desen, and Control) in eight community settings defined in Table 6. These eight community settings are based on combinations of availability of physicians, illness and outbreak probabilities, and compliance and mechanism availability, each of which could be either high or low as defined in Table 5. We run each simulation three times for 10,000 steps, and compute the metrics defined in Section 7.3. We perform a one-way ANOVA to test the null hypotheses H5, H6, and H7. If a null hypothesis is rejected—indicating there is a difference in the solutions, in the post hoc analysis, we perform a series of pairwise *t*-tests to isolate the differences. To avoid a type-1 error, we apply the Bonferroni correction and set the significance cut-off at 0.00833 for H5 and 0.005 for H6 and H7.

Table 6. Community settings.

#	Physicians	Illness probability	Outbreak probability	Compliance[†]
1	low	low	low	high
2	low	high	high	high
3	high	low	low	high
4	high	high	high	high
5	low	low	low	low
6	low	high	high	low
7	high	low	low	low
8	high	high	high	low

[†] For NORM, STOS, DESEN and Control: External physician compliance;
For MECH: Mechanism availability

Figures 10 and 11 in Appendix H show the plots for life sustenance and social welfare for the five solutions in the eight community settings; respectively. Table 7 summarizes the results. The One-way ANOVA test rejects the null hypotheses H5 ($p = 0.025; \eta^2 = 0.28$), H6 ($p = 0.019; \eta^2 = 0.28$), and H7 ($p = 0.03; \eta^2 = 0.27$) with large effect size [Grissom and Kim 2012]. This indicates that there is a difference in the disclosure, sustenance, welfare provided by the five solutions.

In the plots and results summary table, we observe that DESEN yields better average life sustenance and average social welfare compared to the benchmark solutions and the Control solution from the modeler study. Specifically, DESEN yields best life sustenance and social welfare in three of the four communities with low external physician compliance and second best in three other communities having either both physicians and compliance as high or both as low. Tables 11 and 12 in Appendix H list these values yielded by the five solutions for each of the eight communities. We attribute better sustenance and welfare yielded by DESEN to the additional commitment on physicians to share PHI with family during emergency. Recall that this commitment enables physician agents to behave in a proactive manner in emergencies. STOS yields average life sustenance and average social welfare values second only to those yielded by DESEN. MECH yields the lowest sustenance and welfare. The low sustenance and welfare values in MECH are because of catastrophic failures resulting due to mechanism nonavailability when treating a critically ill patient. Such catastrophic failures do not occur with other solutions (note that the mechanism in STOS that is used to prevent physicians from publishing PHI online does not have an effect on patient health). Although we observe differences in the means, in the post hoc analysis, we find that the differences in sustenance and welfare are not significant.

We further observe that STOS results in significantly lower unauthorized information disclosure than NORM and Control. We attribute the lower disclosure in STOS to the mechanism that prevents physicians from publishing PHI online. DESEN results in lower (but not significant) disclosure than NORM and Control in six of the eight communities. Whereas the disclosure in STOS is lower than that in DESEN, the difference is not statistically significant considering the Bonferroni correction. Table 10 (in Appendix H) summarizes the unauthorized disclosure results.

Table 7. Comparing average information disclosure, average life sustenance, and average social welfare across communities for the five solutions. Bold indicates better.

Solution	Norm	Mech	Stos	Desen	Control
Information disclosure	0.97	–	**0.19**	0.95	0.97
Life sustenance	56.91	34.48	59.15	**61.71**	57.63
Social welfare	44.97	28.11	46.65	**48.75**	45.55

7.6 Threats, Mitigation, and Assumptions

Since we simulate resident illnesses, disease outbreaks, physician compliance, and mechanism availability, a threat is whether the values for these variables seeded in the simulation are resonable. To mitigate this threat of reliability of data, we define multiple community settings to simulate different contexts.

Our community healthcare simulation environment relies on a number of simplifying assumptions. First, we assume that all physicians are equally compliant. In reality, different physicians may have different compliance. Second, we assume that all residents are equally susceptible to illness, but in reality residents could have different immunity and thus, may not be equally susceptible to a disease. Third, we assume only two types of illnesses—critical and noncritical, each taking a fixed time to treat. In reality, there could be multiple type of illnesses, each with different treatment times. We make these assumptions because our objective is not to model healthcare reality but to compare the effectiveness of the five solutions.

8 RELATED WORK

We review relevant approaches to Desen in the context of STSs. Chopra et al. [2014] propose a formalization for requirements engineering of sociotechnical systems via social refinement rules. Sommerville et al. [2012] describe sociotechnical systems as users, processes, and technological systems. Process definitions outline how system designers intend the system should be used. In practice, users interpret and adapt them in unpredictable ways. We regulate compliance with such processes via the notion of social norms as well as incorporate agent autonomy into the processes, which constitute the center of our approach. Prior formulations treat norms as expected social properties [Criado et al. 2013], usually enforced through (positive or negative) social sanctions [Nardin et al. 2016]. Criado et al. [2013] propose the normative architecture MaNEA as a coordination mechanism for open multiagent systems, where agents may not be equipped with normative reasoning capabilities. They investigate previous norm enforcement approaches, and describe the challenges of adapting such formalizations to represent real domains. In Desen, we provide formal relations accompanying norm specifications, formalize additional STS components, and demonstrate how transformations between the technical and social tiers can be performed using practical patterns.

Sergot [2013] discusses the correspondence of normative relations among agents with Hohfeldian legal concepts such as duties and rights. Alechina et al. [2013] focus on conditional norms with deadlines, and extend CTL and ATL with sanctions to reason about the effects of normative update. They measure norm compliance by verifying if specific states are reached before the deadline, and enforce norms via sanctions. King et al. [2015] propose a hierarchical governance model for institutions using a multitier normative system using answer set programming. Institutions on the higher level of hierarchy govern others on lower levels. Enhancing Desen with additional normative concepts such as sanctions would provide alternative means of refinement from mechanisms to norms.

Barth et al. [2006] 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 [2013], 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. DESEN regulates interactions between agents, and agents and other components using a richer normative formalization. Moreover, we provide realistic healthcare scenarios that support openness, where agents enter or leave the system dynamically (e.g., recruiting outside physicians in emergencies). Our design patterns capture transitions among the tiers to provide flexibility to agents and to enable secure collaboration.

Agent-oriented modeling techniques have been employed to aid software engineering. Chopra and Singh [2016b] introduce the Interaction-Oriented Software Engineering (IOSE) methodology to capture agent autonomy in development of sociotechnical systems. They identify the limitations of machine-only specifications, and formalize social protocols via norms as we do in this paper. Having a formal social protocol enables compliance checking. Chopra and Singh define five core principles of IOSE: accountability modularity, abstraction, separation of technical and social concerns, encapsulation, and configuration, and evaluate other software engineering methodologies with respect to those principles. DESEN is compatible with IOSE and enables the transition between technical and social components of STSs via the use of patterns.

Conflicts in normative systems are discussed in works related to norm synthesis, wherein conflicting norms are synthesized to nonconflicting abstract norms. Günay and Yolum [2013] discuss the feasibility of commitments, i.e., whether it is possible to satisfy all (existing and prospective) commitments of an agent. They formulate feasibility as a constraint satisfaction problem. Vasconcelos et al. [2009] propose methods for resolving conflicts among norms. Ajmeri et al. [2016] propose Coco, a formalism to express and reason about conflicting commitment instances at runtime, and dominance among them. Coco employs Answer Set Programming to compute nondominated commitment instances and uses Alechina et al.'s [2013] framework to determine compliance of actions with nondominated commitment instances. In contrast, DESEN not only handles commitments, but provides a richer notion of norms and normative design patterns for flexible specifications.

9 CONCLUSIONS

We proposed DESEN, a formal model for STS based on norms that supports requirements verification via model checking and refinement via normative patterns. The main contribution is a way to incorporate STS considerations in AOSE, which has not been explored in depth by previous work. Our design patterns provide means for transforming STS specifications between the technical and social tiers, and exploring the tradeoffs among them [Kafalı et al. 2017a]. Our findings suggest that DESEN's patterns are helpful for participants who are inexperienced in conceptual modeling and norms.

Our work opens up interesting directions for enhancements. We have developed generic patterns, and although we have validated their usability, we have not investigated their completeness. Whereas our norm refinement patterns are generic, they are inspired by healthcare law and practice, and they do not constitute an exhaustive collection. It would help to investigate domain-specific patterns (e.g., switching to priority norms for life threatening situations). Sanctions [Nardin et al. 2016] would add another dimension to DESEN's set of normative patterns. Sanctions provide deterrence against and compensation for norm violations. Further investigation of possible norm violation paths alongside consideration of sanctions would help prevent misuse, i.e., faults caused by autonomous parties [Kafalı et al. 2016b]. Moreover, automated mining of norms [Avery et al.

2016; Dam et al. 2015] would enhance the planning process that we have proposed for generating refined specifications.

Our language can be extended to include instances of agents at design time, which would be helpful for setting up norms and assumptions regarding individual agents. For example, John might be a trustworthy physician agent whereas this assumption cannot be generalized to all other agents pertaining to the physician role. Therefore, additional norms or mechanisms might be needed to regulate agents other than John.

Our simulation environment can be extended to identify tradeoff and conflict scenarios and optimize resources in medical emergencies. Several works deal with normative conflicts [Ajmeri et al. 2016; Günay and Yolum 2013; Vasconcelos et al. 2009] and those could be incorporated in Desen. Such scenarios can also be incorporated in a follow-up human participant study with additional tool support. The development of comprehensive design tool support is left for future work.

Regarding the scalability and selection of patterns, our simulation environment can be extended in a way that the metrics we have provided act as the basis for heuristic optimization. For example, simulation results can provide guidance on the optimal number of outside physicians or maximum tolerable fault rates for mechanisms. Such optimization would guide the design of STS by determining which patterns to apply, e.g., apply normify if high mechanism fault rates cannot be tolerated. The implementation of such an optimization tool is left for future work.

We performed an end to end evaluation of the five methodological steps described in the "Practical usage and implications" section. Specifically, we validated the model checking tool (Step i) and the refinement patterns (Step iv) on a realistic use case, and evaluated the solutions gathered from our modeler study using the simulator (Steps ii and iii). Further validation of these steps is left for future work as part of a larger study with practitioners.

ACKNOWLEDGMENTS

This research was supported by the US Department of Defense under the Science of Security Lablet grant to NC State University.

REFERENCES

Yoosef Abushark, John Thangarajah, Tim Miller, James Harland, and Michael Winikoff. 2015. Early Detection of Design Faults Relative to Requirement Specifications in Agent-Based Models. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1071–1079.

ACEP. 2013. Guidelines for Crisis Standards of Care during Disasters. (2013). American College of Emergency Physicians (ACEP). http://goo.gl/HXWRnH.

Nirav Ajmeri, Hui Guo, Pradeep K. Murukannaiah, and Munindar P. Singh. 2018. Robust Norm Emergence by Revealing and Reasoning about Context: Socially Intelligent Agents for Enhancing Privacy. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Stockholm, 28–34.

Nirav Ajmeri, Jiaming Jiang, Rada Y. Chirkova, Jon Doyle, and Munindar P. Singh. 2016. Coco: Runtime Reasoning about Conflicting Commitments. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, New York, 17–23.

Natasha Alechina, Mehdi Dastani, and Brian Logan. 2013. Reasoning about Normative Update. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Beijing, 20–26.

Paul C. Attie, Munindar P. Singh, Amit P. Sheth, and Marek Rusinkiewicz. 1993. Specifying and Enforcing Intertask Dependencies. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, Dublin, 134–145.

Daniel Avery, Hoa K. Dam, Bastin T. R. Savarimuthu, and Aditya K. Ghose. 2016. Externalization of Software Behavior by the Mining of Norms. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR)*. ACM, Austin, Texas, 223–234.

Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. 2006. Privacy and Contextual Integrity: Framework and Applications. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, 184–198.

Jaspreet Bhatia, Travis D. Breaux, and Florian Schaub. 2016. Mining privacy goals from privacy policies using hybridized task recomposition. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 25, 3 (2016), 22.

Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. 2004. Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 8, 3 (May 2004), 203–236.

Amit K. Chopra, Fabiano Dalpiaz, F. Başak Aydemir, Paolo Giorgini, John Mylopoulos, and Munindar P. Singh. 2014. Protos: Foundations for Engineering Innovative Sociotechnical Systems. In *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*. IEEE Computer Society, Karlskrona, Sweden, 53–62.

Amit K. Chopra and Munindar P. Singh. 2009. Multiagent Commitment Alignment. In *Proceedings of the 8th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Budapest, 937–944.

Amit K. Chopra and Munindar P. Singh. 2015. Cupid: Commitments in Relational Algebra. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*. AAAI Press, Austin, Texas, 2052–2059.

Amit K. Chopra and Munindar P. Singh. 2016a. Custard: Computing Norm States over Information Stores. In *Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Singapore, 1096–1105.

Amit K. Chopra and Munindar P. Singh. 2016b. From Social Machines to Social Protocols: Software Engineering Foundations for Sociotechnical Systems. In *Proceedings of the 25th International World Wide Web Conference*. ACM, Montréal, 903–914.

Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. 2002. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of the International Conference on Computer Aided Verification (CAV) (LNCS)*. Springer-Verlag, London, 359–364.

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 1999. *Model Checking*. MIT Press, Cambridge, MA.

Natalia Criado, Estefania Argente, Pablo Noriega, and Vicente Botti. 2013. MaNEA: A Distributed Architecture for Enforcing Norms in Open MAS. *Engineering Applications of Artificial Intelligence* 26, 1 (Jan. 2013), 76–95.

Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. 2013. Adaptive socio-technical systems: A requirements-based approach. *Requirements engineering* 18, 1 (2013), 1–24.

Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. 2016. *Security Requirements Engineering: Designing Secure Socio-Technical Systems*. The MIT Press, Cambridge, MA.

Hoa Khanh Dam, Bastin Tony Roy Savarimuthu, Daniel Avery, and Aditya K. Ghose. 2015. Mining Software Repositories for Social Norms. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Florence, Italy, 627–630. New Ideas and Emerging Results Track.

Robert Darimont and Axel Van Lamsweerde. 1996. Formal refinement patterns for goal-driven requirements elaboration. *ACM SIGSOFT Software Engineering Notes* 21, 6 (1996), 179–190.

Mehdi Dastani, Koen V. Hindriks, and John-Jules C. Meyer. 2010. *Specification and Verification of Multi-agent Systems* (1st ed.). Springer, New York.

Renzo Degiovanni, Dalal Alrajeh, Nazareno Aguirre, and Sebastian Uchitel. 2014. Automated Goal Operationalisation Based on Interpolation and SAT Solving. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. ACM, New York, NY, USA, 129–139.

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1998. Property specification patterns for finite-state verification. In *Proceedings of the Second workshop on Formal methods in software practice*. ACM, New York, NY, USA, 7–15.

Predrag Filipovikj, Guillermo Rodriguez-Navas, Mattias Nyberg, and Cristina Seceleanu. 2018. Automated SMT-based consistency checking of industrial critical requirements. *ACM SIGAPP Applied Computing Review* 17, 4 (2018), 15–28.

Robin A. Gandhi and Seok W. Lee. 2011. Discovering multidimensional correlations among regulatory requirements to understand risk. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20, 4 (2011), 16.

Jorge J. Gómez-Sanz and Rubén Fuentes-Fernández. 2015. Understanding Agent-Oriented Software Engineering methodologies. *The Knowledge Engineering Review* 30, 4 (Sep 2015), 375–393.

Robert J. Grissom and John J. Kim. 2012. *Effect Sizes for Research: Univariate and Multivariate Applications*. Routledge, Abingdon-on-Thames.

Akın Günay, Michael Winikoff, and Pınar Yolum. 2015. Dynamically generated commitment protocols in open systems. *Autonomous Agents and Multi-Agent Systems* 29, 2 (2015), 192–229.

Akın Günay and Pınar Yolum. 2013. Constraint satisfaction as a tool for modeling and checking feasibility of multiagent commitments. *Applied Intelligence* 39, 3 (Oct. 2013), 489–509.

Bronwyn H. Hall. 2004. *Innovation and Diffusion*. Technical Report. National Bureau of Economic Research.

Christopher Ham, Robert Dingwall, Paul Fenn, and Don Harris. 1988. *Medical Negligence: Compensation and Accountability.* Centre for Socio-Legal Studies, King's Fund Institute, London.

Donna Hammaker. 2010. *Health Care Management and the Law: Principles and Applications.* Cengage Learning, Boston.

HHS. 2014. Bulletin: HIPAA Privacy in Emergency Situations. (2014). United States Department of Health and Human Services (HHS). http://www.hhs.gov/ocr/privacy/hipaa/understanding/special/emergency/.

HHS. 2019. Breach Portal: Notice to the Secretary of HHS Breach of Unsecured Protected Health Information Affecting 500 or More Individuals. (2019). United States Department of Health and Human Services (HHS). https://ocrportal.hhs.gov/ocr/breach/.

Myles Hollander and Douglas A. Wolfe. 1999. *Nonparametric Statistical Methods.* Wiley, New York.

Jennifer Horkoff and Eric Yu. 2016. Interactive Goal Model Analysis for Early Requirements Engineering. *Requirements Engineering* 21, 1 (March 2016), 29–61.

Andrew J. I. Jones and Marek J. Sergot. 1993. On the Characterisation of Law and Computer Systems: The Normative Systems Perspective. In *Deontic Logic in Computer Science: Normative System Specification*, John-Jules Ch. Meyer and Roel J. Wieringa (Eds.). John Wiley and Sons, Chichester, United Kingdom, Chapter 12, 275–307.

Özgür Kafalı, Nirav Ajmeri, and Munindar P. Singh. 2016a. Revani: Revising and Verifying Normative Specifications for Privacy. *IEEE Intelligent Systems* 31, 5 (Sept. 2016), 8–15.

Özgür Kafalı, Nirav Ajmeri, and Munindar P. Singh. 2017a. Kont: Computing Tradeoffs in Normative Multiagent Systems. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI).* AAAI, San Francisco, 3006–3012.

Özgür Kafalı, Jasmine Jones, Megan Petruso, Laurie Williams, and Munindar P. Singh. 2017b. How Good is a Security Policy against Real Breaches? A HIPAA Case Study. In *Proceedings of the 39th International Conference on Software Engineering (ICSE).* IEEE Computer Society, Buenos Aires, 530–540.

Özgür Kafalı, Munindar P. Singh, and Laurie Williams. 2016b. Nane: Identifying Misuse Cases Using Temporal Norm Enactments. In *Proceedings of the 24th IEEE International Requirements Engineering Conference (RE).* IEEE Computer Society, Beijing, 136–145.

Geylani Kardas. 2013. Model-driven development of multiagent systems: a survey and evaluation. *The Knowledge Engineering Review* 28 (12 2013), 479–503.

Thomas C. King, Tingting Li, Marina De Vos, Virginia Dignum, Catholijn M. Jonker, Julian Padget, and B. Birna van Riemsdijk. 2015. A Framework for Institutions Governing Institutions. In *Proceedings of the 14th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS).* IFAAMAS, Istanbul, 473–481.

Nadin Kökciyan and Pınar Yolum. 2016. PriGuard: A Semantic Approach to Detect Privacy Violations in Online Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (Oct 2016), 2724–2737.

Ross Koppel, Sean W. Smith, Jim Blythe, and Vijay Kothari. 2015. Workarounds to Computer Access in Healthcare Organizations: You Want My Password or a Dead Patient? *Studies in health technology and informatics* 208 (2015), 215–220.

Christian Kuster, Tobias Küster, Marco Lützenberger, and Sahin Albayrak. 2014. Model-driven Development and Validation of Multi-agent Systems in JIAC V with the Agent World Editor. *Procedia Computer Science* 32 (2014), 920–927.

Emmanuel Letier and William Heaven. 2013. Requirements Modelling by Synthesis of Deontic Input-output Automata. In *Proceedings of the 35th International Conference on Software Engineering (ICSE).* IEEE Press, Piscataway, NJ, USA, 592–601.

Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. 2005. MASON: A Multiagent Simulation Environment. *Simulation: Transactions of the Society for Modeling and Simulation International* 81, 7 (July 2005), 517–527.

Luis G. Nardin, Tina Balke-Visser, Nirav Ajmeri, Anup K. Kalia, Jaime S. Sichman, and Munindar P. Singh. 2016. Classifying sanctions and designing a conceptual sanctioning process model for socio-technical systems. *The Knowledge Engineering Review* 31 (March 2016), 142–166. Issue 2.

H. Van Dyke Parunak and Sven A. Brueckner. 2015. Software engineering for self-organizing systems. *The Knowledge Engineering Review* 30, 4 (Sep 2015), 419–434.

Klaus Pohl. 1994. The Three Dimensions of Requirements Engineering: A Framework and Its Applications. *Information Systems* 19, 3 (April 1994), 243–258.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Singapore.

Marek Sergot. 2013. Normative Positions. In *Handbook of Deontic Logic and Normative Systems*, Dov Gabbay, John Horty, Ron van der Meyden, Xavier Parent, and Leendvert van der Torre (Eds.). College Publications, London, Chapter 5, 353–406.

Munindar P. Singh. 2011. Trust as Dependence: A Logical Approach. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS).* IFAAMAS, Taipei, 863–870.

Munindar P. Singh. 2013. Norms as a Basis for Governing Sociotechnical Systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 1 (Dec. 2013), 21:1–21:23.

Munindar P. Singh, Amit K. Chopra, and Nirmit Desai. 2009. Commitment-Based Service-Oriented Architecture. *IEEE Computer* 42, 11 (Nov. 2009), 72–79.

Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid, and Richard Paige. 2012. Large-Scale Complex IT Systems. *Communications of the ACM (CACM)* 55, 7 (July 2012), 71–77.

Jose M. Such and Natalia Criado. 2016. Resolving multi-party privacy conflicts in social media. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1851–1863.

Tavneet Suri. 2011. Selection and Comparative Advantage in Technology Adoption. *Econometrica* 79, 1 (2011), 159–209.

Christos Tsigkanos, Liliana Pasquale, Claudio Menghi, Carlo Ghezzi, and Bashar Nuseibeh. 2014. Engineering topology aware adaptive security: Preventing requirements violations at runtime. In *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*. IEEE, Karlskrona, Sweden, 203–212.

Wamberto W. Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. 2009. Normative Conflict Resolution in Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems* 19, 2 (Oct. 2009), 124–152.

Georg H. Von Wright. 1999. Deontic Logic: A Personal View. *Ratio Juris* 12, 1 (1999), 26–38.

Pamela Zave and Michael Jackson. 1997. Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology* 6, 1 (Jan 1997), 1–30.

Hong Zhu and Ian Bayley. 2013. An algebra of design patterns. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 22, 3 (2013), 23.

A NUSMV MODEL

We describe the NuSMV model for the HIPAA privacy scenario. A NuSMV main module begins with the variable declarations as shown in Listing 3. These variables constitute the elements of a state in NuSMV. They can be defined as Boolean variables (Lines 5–16 and 30–34), enumerations (Line 18–21), or as instances of norms (Lines 23–28).

Listing 3. Variable declarations in NuSMV.

```
1  MODULE main

3  VAR

5  confidential_communication: boolean;
6  national_emergency: boolean;
7  emergency_area: boolean;
8  emergency_period: boolean;
9  president_declare: boolean;
10 secretary_HHS_declare: boolean;
11 patient_consent: boolean;
12 share_PHI_thirdparty: boolean;
13 share_PHI_outside_physician: boolean;
14 share_PHI_family: boolean;
15 ...
16 non_share_PHI_family: boolean;--complement of share_PHI_family

18 r1: {PHY, HOS, PAT, OUTPHY};
19 r2: {PHY, HOS, PAT, OUTPHY};
20 r3: {PHY, HOS, PAT, OUTPHY};
21 r4: {PHY, HOS, PAT, OUTPHY};

23 c1: c(r1, r3, confidential_communication & !national_emergency,
24              accommodate_request, FALSE);
25 a1: a(r1, r2, TRUE, confidential_communication, FALSE);
26 p1: p(r1, r2, !patient_consent, share_PHI_thirdparty, FALSE);
27 p2: p(r1, r2, !patient_consent, share_PHI_outside_physician, FALSE);
28 p3: p(r1, r2, !patient_consent, share_PHI_family, FALSE);

30 sat_c1: boolean;
31 sat_a1: boolean;
32 sat_p1: boolean;
33 sat_p2: boolean;
34 sat_p3: boolean;
```

Listing 4 describes the initial state in NuSMV by assigning values to the Boolean variables (Lines 1–12 and 19–23) and enumerations (Line 14–17). Note that the agent roles for the specified norms do not change during execution. We ensure this via next state declarations (Line 25–28).

Listing 5 describes the state transition rules in NuSMV with respect to the stated mechanisms. Mechanism m_1 controls the transition rules for emergency. Accordingly, a national emergency is declared when the president and the secretary of HHS declare an emergency situation (Line 4).

Mechanism m_2 controls the transition rules for consent. Accordingly, a patient's consent is not valid if the the patient requests confidential communications (Line 11).

Listing 4. Variable assignments in NuSMV.

```
1  init(confidential_communication):= FALSE;
2  init(national_emergency):= FALSE;
3  init(emergency_area):= FALSE;
4  init(emergency_period):= FALSE;
5  init(president_declare):= FALSE;
6  init(secretary_HHS_declare):= FALSE;
7  init(patient_consent):= FALSE;
8  init(share_PHI_thirdparty):= FALSE;
9  init(share_PHI_outside_physician):= FALSE;
10 init(share_PHI_family):= FALSE;
11 ...
12 init(non_share_PHI_family):= FALSE;

14 init(r1):= PHY;
15 init(r2):= HOS;
16 init(r3):= PAT;
17 init(r4):= OUTPHY;

19 init(sat_c1):= FALSE;
20 init(sat_a1):= FALSE;
21 init(sat_p1):= FALSE;
22 init(sat_p2):= FALSE;
23 init(sat_p3):= FALSE;

25 next(r1):= PHY;
26 next(r2):= HOS;
27 next(r3):= PAT;
28 next(r4):= OUTPHY;
```

Listing 5. Mechanisms in NuSMV.

```
1  --m1
2  next(national_emergency):=
3   case
4    president_declare & secretary_HHS_declare: {TRUE, FALSE};
5    TRUE: FALSE;
6   esac;

8  --m2
9  next(patient_consent):=
10  case
11   confidential_communication & !national_emergency: FALSE;
12   TRUE: {TRUE, FALSE};
13  esac;
```

Listing 6 describes the state transition rules in NuSMV with respect to the stated norms.

Listing 6. Norms in NuSMV.

```
next(sat_c1):=
 case
  accommodate_request: TRUE;
  TRUE: FALSE;
 esac;

--a1
next(sat_a1):=
 case
  confidential_communication: TRUE;
  TRUE: FALSE;
 esac;

--p1
next(sat_p1):=
 case
  non_share_PHI_thirdparty: TRUE;
 esac;

--p2
next(sat_p1):=
 case
  non_share_PHI_outside_physician: TRUE;
 esac;

--p3
next(sat_p1):=
 case
  non_share_PHI_family: TRUE;
 esac;
```

B ILLUSTRATION OF DSL SYNTAX

We now illustrate the DSL by applying it to the example of Table 3. The code listing includes comments for convenience. We coupled assumptions with their corresponding norms.

Listing 7. DSL for Example 1.

```
// --- ROLES ---
role Hospital
role Patient
role Physician
role Outside_Physician

// --- MECHANISMS ---
// national emergency can be declared when president
// and secretary of health both declare emergency
mechanism m1
 when president_declare && secretary_HHS_declare
 possible add emergency

// a hospital cannot operate in an emergency mode
// if the hospital is not located in the emergency area
// or if it is not currently a period of emergency
assumption s2
 if !emergency_area || !emergency_period
 then !emergency

// patient's consent to share PHI can be overridden when
// patient requests confidential communications during nonemergency
mechanism m2
 when confidential_communication && !emergency
 possible delete consent

// patient can give consent at any time (unconditional)
mechanism m3
 possible add consent

// it is not possible to obtain consent from patients during emergency
assumption s1
 if emergency
 then !consent

// --- NORMS and ASSUMPTIONS ---
// physician is committed to patient to accommodating patient's request
// if patient requests confidential communications during nonemergency
commitment c1 Physician to Patient
 if confidential_communication && !emergency
 must accommodate_request
```

```
// trustworthy physician will not violate c1
assumption s3
 if trustworthy_physician && confidential_communication && !emergency
 then accommodate_request

// physicians are trustworthy (unconditional)
assumption s5
 trustworthy_physician

// patient is authorized by hospital to request confidential
// communications during nonemergency operations
authorization a1 Patient by Hospital
 if !emergency
 allow confidential_communication

// physician is prohibited by hospital from sharing patient's PHI
// with third parties if the patient does not give consent
prohibition p1 Physician by Hospital
 if !consent
 forbid share_PHI_third_party

// physician is prohibited by hospital from sharing patient's PHI with
// outside physician if no consent from patient during nonemergency
prohibition p2 Physician by Hospital
 if !consent && !emergency
 forbid share_PHI_outside_physician

// physician is prohibited by hospital from sharing patient's PHI with
// patient's family if no consent from patient during nonemergency
prohibition p3 Physician by Hospital
 if !consent && !emergency
 forbid share_PHI_family

// outside physician is prohibited by hospital from sharing patient's
// PHI with third parties at all times (unconditional, does not expire)
prohibition p4 Outside_Physician by Hospital
 forbid share_PHI_third_party

// trustworthy outside physician will not violate p4
assumption s4
 if trustworthy_outside_physician
 then !share_PHI_third_party

// outside physicians are trustworthy (unconditional)
assumption s6
 trustworthy_outside_physician
```

C STUDY DESIGN

C.1 Effect of Prior Experience

To evaluate the influence of DESEN on participants with prior experience or no prior experience on producing specifications, we additionally propose subhypotheses for the hypotheses on coverage $H1_{coverage}$, correctness $H2_{correctness}$, and time $H3_{time}$. For each subhypothesis, a null hypothesis (omitted for brevity) states that there is no difference between the groups.

Coverage: The following subhypotheses evaluate if DESEN influences the coverage of specifications produced by participant with prior experience with norms and without prior experience.

H1a DESEN and Control differently influence the coverage of specifications produced by participants with prior industry experience or knowledge of norms.

H1b DESEN and Control differently influence the coverage of specifications produced by participants with no prior industry experience or knowledge of norms.

Correctness: The following subhypotheses evaluate if DESEN influences the correctness of specifications produced by participant with prior experience with norms and without prior experience.

H2a DESEN and Control differently influence the correctness of specifications produced by participants with prior industry experience or knowledge of norms.

H2b DESEN and Control differently influence the correctness of specifications produced by participants with no prior industry experience or knowledge of norms.

Time: The following subhypotheses evaluate if DESEN influences the time expended to produce specifications by participants with prior experience with norms and without prior experience.

H3a DESEN and Control differently influence the time expended to produce specification by participants with prior industry experience or knowledge of norms.

H3b DESEN and Control differently influence the time expended to produce specification by participants with no prior industry experience or knowledge of norms.

D TUTORIALS

Participants in the Control group are given a tutorial on normative requirements including a basic definition of refinement described in Appendix D.1. In addition to the tutorial on normative requirements, participants in the Desen group are given a note on refinement patterns described in Appendix D.2.

D.1 Normative Requirements and Refinement

We outline an approach to capture natural language requirements as social norms. We review the background on norms and requirements, and use examples to illustrate the connection between them.

Social Norms. Humans, organizations and technical systems such as software interplay with each other in a *sociotechnical* system (STS). To capture the requirements of an STS, we adopt Singh's [Singh 2013] model of norms. A norm is directed from a subject to an object and is constructed as a conditional relationship involving an antecedent (which brings the norm in force) and a consequent (which brings the norm to satisfaction). This representation yields clarity on who is accountable to whom. A norm has four core elements—subject, object, antecedent, and consequent. It can be formalized as N(subject, object, *antecedent*, *consequent*).

Norms in our approach are of the following main types.

A commitment means that its subject commits to its object to ensure the consequent if the antecedent holds. For example, in a hospital, physicians are committed to the hospital to operating upon patients when there is an emergency. We write this commitment as:
C(physician, hospital, *emergency*, *operate*)

An authorization means that its subject is authorized by its object for bringing about the consequent if the antecedent holds. For example, physicians are authorized by the patients to access their electronic health record (EHR) if the patients give consent. We write this authorization as:
A(physician, patient, *consent*, *access_EHR*)

A prohibition means that its subject is forbidden by its object from bringing about the consequent if the antecedent holds. For example, healthcare professionals (HCP) are prohibited by the hospital from disclosing patients' protected health information (PHI). We write this prohibition as (true in the antecedent means that the norm is unconditional):
P(hcp, hospital, *true*, *disclose_PHI*)

Requirements. Requirements represent what the stakeholders expect from an STS, and are usually expressed in natural language. Consider the following scenario.

Medical emergency scenario: There has been a public emergency near the hospital, and several unconscious patients need to be operated upon immediately. The hospital does not have the required number of physicians on staff to attend to the emergency situation. Therefore, it has to call in volunteer physicians from nearby hospitals. However, the volunteer physicians are not supposed to disclose the patients' PHI.

There are three requirements associated with the above scenario.

R-Operate Physicians must operate upon patients immediately when there is a medical emergency.

R-Help The hospital may allow volunteer physicians from other hospitals to help with the treatment of patients.

R-Disclose Volunteer physicians must not disclose the patients' PHI.

Refinement. Norm specifications can initially be broad. That is, they may not cover specific situations that might lead to opportunities being missed, and eventually cause violation of some of the requirements. When this is the case, the stakeholders should refine the norms to account for the situation at hand. This is similar to classical refinement, i.e., weakening or strengthening of preconditions or postconditions [Pohl 1994]. Norm refinement can be performed by applying logic operators (NOT, AND, OR) on the propositions of norms (antecedent, consequent). Consider the authorization from earlier: physicians are authorized by the hospital to access patients' EHR provided there is consent. In order to provide additional flexibility to physicians, we can refine this authorization so that physicians are authorized to access patients' EHR provided there is consent or when there is an emergency (i.e., the precondition is weakened).

D.2 Patterns

Patterns provide ways of extending a given norm specification with specific conditions, which enhance flexibility (i.e., enable additional executions of the STS) while preserving functionality.

Relaxation Patterns. The first set of *relaxation* patterns focus on the alteration of antecedents and consequents of norms to promote collaboration among agents. We have the following relaxation patterns.

Release of liability refines a commitment to make its antecedent / consequent more general or more specific. Consider the following commitment:
C(PHYSICIAN, HOSPITAL, *true*, *operate* AND *clinic*): the physician is committed to the hospital to operating upon patients as well as doing clinic duty.
One refinement of the commitment is the following:
C(PHYSICIAN, HOSPITAL, *emergency*, *operate* AND *clinic*): the physician is committed to doing the same tasks only in emergencies (i.e., the antecedent is more specific).
Another refinement of the commitment is the following:
C(PHYSICIAN, HOSPITAL, *true*, *operate*): the physician is only committed to operating upon patients (i.e., the consequent is more general).

Expansion refines an authorization to make its antecedent / consequent more general or more specific. Consider the following authorization:
A(PHYSICIAN, HOSPITAL, *consent*, *own_patients_EHR*): the physician is authorized by the hospital to access her own patients provided there is consent.
One refinement of the authorization is the following:
A(PHYSICIAN, HOSPITAL, *consent*, *own_patients_EHR* OR *other_patients_EHR*): the physician is authorized to access her own patients as well as other patients (i.e., the consequent is more general) provided there is consent.

Accessibility refines a prohibition to make its antecedent / consequent more general or more specific. Consider the following prohibition:
P(PHYSICIAN, HOSPITAL, *true*, *share_PHI_colleague* OR *share_PHI_family*): the physician is prohibited by the hospital from sharing a patient's PHI with colleagues or with the patient's family.
One refinement of the prohibition is the following:
P(PHYSICIAN, HOSPITAL, *true*, *share_PHI_colleague*): the physician is only prohibited from sharing a patient's PHI with colleagues (i.e., the consequent is more specific).

Amendment Patterns. The relaxation patterns provide additional functionality that is not available before. However, they might open up new vulnerabilities if a violation due to the additional functionality is not properly handled. The following *amendment* patterns address these challenges to improve security and privacy related concerns while promoting collaboration.

Responsibility limits the subject of the norm to the intended functionality provided by the relaxation pattern by specifying a complementary commitment. Consider the authorization from the Expansion pattern:
A(PHYSICIAN, HOSPITAL, *consent*, *own_patients_EHR*)
Now, consider the following refinement:
A(PHYSICIAN, HOSPITAL, *consent*, *own_patients_EHR* OR *other_patients_EHR*)
C(PHYSICIAN, HOSPITAL, *own_patients_EHR* OR *other_patients_EHR*, *logout*)
An additional commitment is provided to make sure that the physician logs out from the computer after she finishes reviewing a patient's EHR.

Limitation limits the subject of the relaxed norm by specifying a complementary prohibition. Consider the prohibition from the Accessibility pattern:
P(PHYSICIAN, HOSPITAL, *true*, *share_PHI_colleague* OR *share_PHI_family*)
Now, consider the following refinement:
P(PHYSICIAN, HOSPITAL, *true*, *share_PHI_colleague*)
P(PHYSICIAN, HOSPITAL, *true*, *publish_PHI_online*)
An additional prohibition is provided to make sure that the physician does not publish a patient's PHI online.

E SURVEYS

E.1 Pre-participation Survey

To select participant for the study, we circulated an advertisement with a link to a pre-participation survey. We received 68 responses of which we selected 32 computer science graduate students, and created groups balanced in terms of familiarity and industry experience.

(1) Which degree are you currently pursuing? (e.g., BS, MS, PhD.)
(2) What is your current major? (e.g., Computer Science, Computer Networking, Electrical and Computer Engineering.)
(3) How long is your academic (programming and software development) experience?
(4) How long is your industry work experience?
(5) How familiar are you with conceptual modeling?
(6) How familiar are you with norms?
(7) How familiar are you with security and privacy in the healthcare domain?
(8) How familiar are you with role based access controls?

E.2 Post Survey

(1) How easy was it to understand the problem domain?
(2) How easy were the scenarios?
(3) To what extent the patterns helped you in refinement? *
(4) To what extent a refinement tool listing all possible applications of the patterns would have helped you? *
(5) To what extent more guidance on patterns would have helped in refinement? #

*Answered only by participants in Desen group. #Answered only by participants in Control group.

F SCENARIOS AND DELIVERABLES

F.1 Healthcare Privacy Scenario

Consider a healthcare scenario involving the privacy of patient's protected health information.

Hospitals are bound by law to keep their patient's electronic health records (EHR) private. Therefore, when a patient is admitted to a hospital, the physician treating the patient must not publish the patient's protected health information (PHI) online. However, the physician may share the patient's PHI with a colleague to get an expert opinion. Moreover, the physician may share the patient's PHI with the patient's family when there is an emergency.

Agents

PHYSICIAN, PATIENT, HOSPITAL

Propositions

$true, emergency, patient\_visit, access\_ehr, publish\_phi\_online, share\_phi\_colleague, share\_phi\_family$

Deliverables

- Specify the natural language requirements for the scenario
- A requirements analyst has come up with the following norms for the scenario:
  P(PHYSICIAN, HOSPITAL, $true, publish\_phi\_online$ OR $share\_phi\_colleague$)
  A(PHYSICIAN, HOSPITAL, $true$, $share\_phi\_family$)
  Your task is to come up with a refinement of this specification using the above agents and propositions, so as to satisfy the requirements for the scenario.

F.2 Healthcare Security Scenario

Consider a healthcare scenario involving the security of patient's electronic health records (EHR).

Hospitals are bound by law to keep their patient's electronic health records (EHR) secure. Therefore, healthcare workers who have access to patients' EHR must not share their credentials (username and password) with anyone. If healthcare workers need to use a public computer (such as the computer in the emergency department), they must log off from the computer as soon as they are finished reviewing the patient's EHR. Moreover, healthcare workers must not view the EHR of their friends unless they are responsible for their treatment.

Agents

WORKER, PHYSICIAN, PATIENT, HOSPITAL

Propositions

$true, emergency, share\_id, share\_password, access\_ehr, public\_computer, logout, friend, treating$

Deliverables

- Specify the natural language requirements for the scenario
- A requirements analyst has come up with the following norms for the scenario:
  P(WORKER, HOSPITAL, $true$, $share\_id$ OR $share\_password$)
  C(WORKER, HOSPITAL, $true$, $logout$)
  P(WORKER, HOSPITAL, $true$, $access\_EHR$)
  Your task is to come up with a refinement of this specification using the above agents and propositions, so as to satisfy the requirements for the scenario.

F.3 Academic Access Control Scenario

Consider an access control scenario for an academic building involving the confidentiality and integrity of its sensitive resources.

An academic department has a number of functions to keep running. Exams are kept in the department's safe room. Professors have access to the safe room. Teaching assistants may access exams, however they must not be in the safe room when the professor is present to keep the safe's security code confidential. Exams are printed using the departmental printer. When there is a problem, a technician is called to repair the printer. A visiting technician is allowed to enter the printer room. In general, visitors are only allowed in public areas and rooms they are supposed to carry out work. Often, grad students need training on the department's server. However, they are not allowed to enter the server room when authorized staff are not present to preserve the integrity of the server.

Agents

TECHNICIAN, GRADUATE_STUDENT, TEACHING_ASSISTANT, PROFESSOR, SECURITY_ADMIN

Propositions

$true, printer\_broken, access\_printer, access\_server, access\_safe, access\_public\_area, exam\_period, staff\_present, professor\_present$

Deliverables

- Identify the natural language requirements for the scenario
- A requirements analyst has come up with the following norms for the scenario:
  A(TECHNICIAN, SECURITY_ADMIN, $printer\_broken$, $access\_printer$)
  P(GRADUATE_STUDENT, SECURITY_ADMIN, $true$, $access\_server$)
  A(PROFESSOR, SECURITY_ADMIN, $true$, $access\_safe$)
  P(TEACHING_ASSISTANT, SECURITY_ADMIN, $exam\_period$, $access\_safe$)
  Your task is to come up with a refinement of this specification using the above agents and propositions, so as to satisfy the requirements for the scenario.

G STUDY ANALYSIS

As participants produced specifications for the scenarios each phase, we recorded their completion time. Later, participants' specifications were compared against the oracle solution to compute the coverage and correctness of the specifications produced for each of the scenarios. Table 8 summarizes the results for all participants, participants with no prior knowledge of norms, and participants with no industry experience.

Table 8. Empirical results on effectiveness of Desen compared to Control for all participants, participants with no knowledge of norms and participants with no industry experience in conceptual modeling.

	All participants			No knowledge of norms			No industry experience		
	Desen	Control	*p* value	Desen	Control	*p* value	Desen	Control	*p* value
Time–Phase 1 (in min)	30.63	23.75	<0.05	36.00	25.33	<0.05	35.20	18.40	<0.01
Time–Phase 2 (in min)	15.63	14.88	0.71	15.50	18.00	0.47	12.80	11.60	0.51
Time–Phase 3 (in min)	16.69	19.63	0.15	15.50	18.83	0.05	18.40	18.40	1
Coverage of specification (in %)	65	62	0.66	75	61	0.15	76	56	0.09
Correctness of specification (in %)	67	62	0.55	78	58	<0.05	79	57	<0.05

After completing each phase, participants completed a post-survey. Table 9 summarizes the responses received from the post-survey.

Table 9. Empirical results on effectiveness of Desen based on the post-survey responses.

	Desen		Control	
	Mean	Median	Mean	Medium
How easy was it to understand the problem domain? *	2.60	3	2.60	3
How easy was it to learn the approach? *	2.40	2	2.27	2
How easy was it to define a specification using the approach? *	2.80	3	2.60	3
How easy was it to maintain a specification using the approach? *	2.60	3	2.80	3
How easy were the scenarios? [Phase 1] *	2.27	2	2.00	2
How easy were the scenarios? [Phase 2] *	2.40	2	2.47	2
How easy were the scenarios? [Phase 3] *	3.33	4	3.13	3
To what extent the patterns helped you in refinement? #	4.40	4	-	-
To what extent the more guidance on patterns would have helped in refinement? †	-	-	3.87	4
To what extent a refinement tool listing all possible applications of the patterns would have helped you? †	4.27	4	-	-

*(1: Very easy, 2: Easy, 3: Medium, 4: Difficult, 5: Very difficult)
#(1: Didn't help me at all, 5: Helped a lot)
†(1: No help at all, 5: Would have helped a lot)

H SIMULATION RESULTS

Tables 10, 11, and 12 lists the information disclosure resulted in and life sustenance and social welfare yielded by the five solutions in eight communities.

Table 10. Information disclosure resulted in five solutions for the eight communities. STOS consistently results in lower disclosure. DESEN results in lower disclosure than NORM and MECH in six of the eight communities. Bold indicates best; *Slanted* indicates second best.

	NORM	MECH	STOS	DESEN	Control
Community 1	0.37	–	**0.09**	*0.28*	0.35
Community 2	1.01	–	**0.20**	*0.77*	0.92
Community 3	0.58	–	**0.15**	*0.46*	0.58
Community 4	1.29	–	**0.24**	*1.19*	1.49
Community 5	0.35	–	**0.09**	*0.27*	0.39
Community 6	1.56	–	**0.26**	1.83	*1.17*
Community 7	0.60	–	**0.15**	*0.49*	0.63
Community 8	*2.02*	–	**0.30**	2.25	2.20
Mean	0.97	–	**0.19**	*0.94*	0.97

Table 11. Life sustenance yielded by the five solutions for the eight communities. DESEN yields best sustenance in three of the four communities with low compliance, and and yields second-best sustenance in three other communities when both physicians and compliance are either high or are low. Bold indicates best; *Slanted* indicates second best.

	NORM	MECH	STOS	DESEN	Control
Community 1	**31.53**	28.15	*31.21*	31.16	29.85
Community 2	*44.84*	13.24	**50.98**	41.04	40.41
Community 3	49.24	49.01	**52.10**	*51.43*	49.57
Community 4	56.74	24.98	60.97	*63.33*	**66.45**
Community 5	29.33	25.61	29.05	*30.53*	**32.07**
Community 6	39.95	13.07	*41.43*	**49.40**	33.28
Community 7	50.61	47.76	50.53	**54.32**	*52.52*
Community 8	57.52	23.10	56.92	**68.78**	*60.25*
Mean	44.97	28.11	*46.65*	**48.75**	45.55

Figures 10 and 11 show the plots for life sustenance and social welfare for the five solutions in the eight community settings; respectively.

Figures 12, 13, and 14 show the box plots for information disclosure, life sustenance, and social welfare in the five solutions across the eight community settings; respectively.

Table 12. Social welfare yielded by the five solutions for the eight communities. Desen yields best sustenance in three of the four communities with low compliance, and yields second-best welfare in three other communities when both physicians and compliance are either high or are low. **Bold** indicates best; *Slanted* indicates second best.

	Norm	Mech	Stos	Desen	Control
Community 1	**36.40**	32.53	*35.98*	35.91	34.45
Community 2	*61.01*	18.36	**69.35**	55.69	55.01
Community 3	56.54	56.32	**59.89**	*58.90*	56.92
Community 4	77.01	34.24	82.68	*85.68*	**90.34**
Community 5	33.84	29.63	33.55	*35.18*	**36.95**
Community 6	54.35	18.10	*56.35*	**67.01**	45.35
Community 7	58.15	54.92	58.06	**62.29**	*60.30*
Community 8	78.01	31.73	77.34	**93.00**	*81.67*
Mean	56.91	34.48	*59.15*	**61.71**	57.63

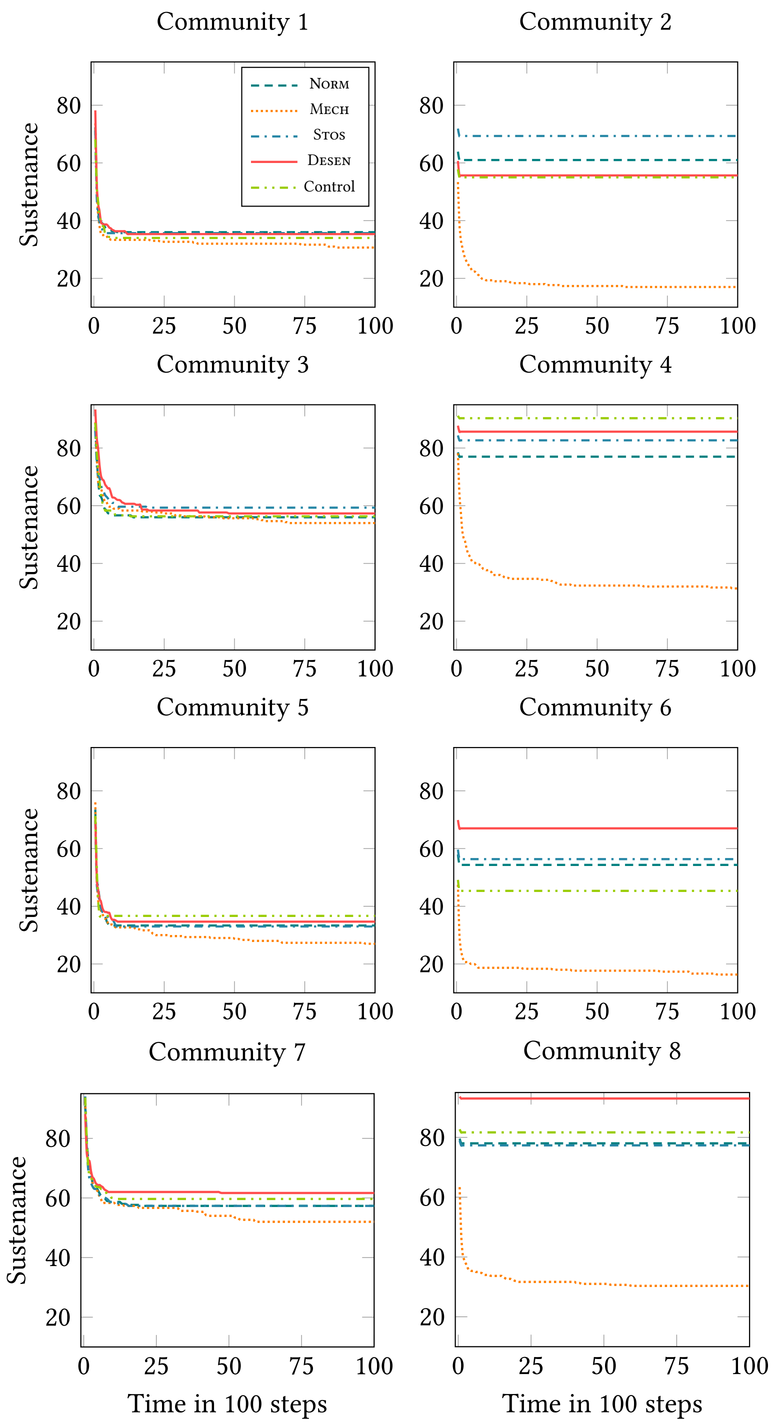


Fig. 10. Life sustenance in the five solutions.

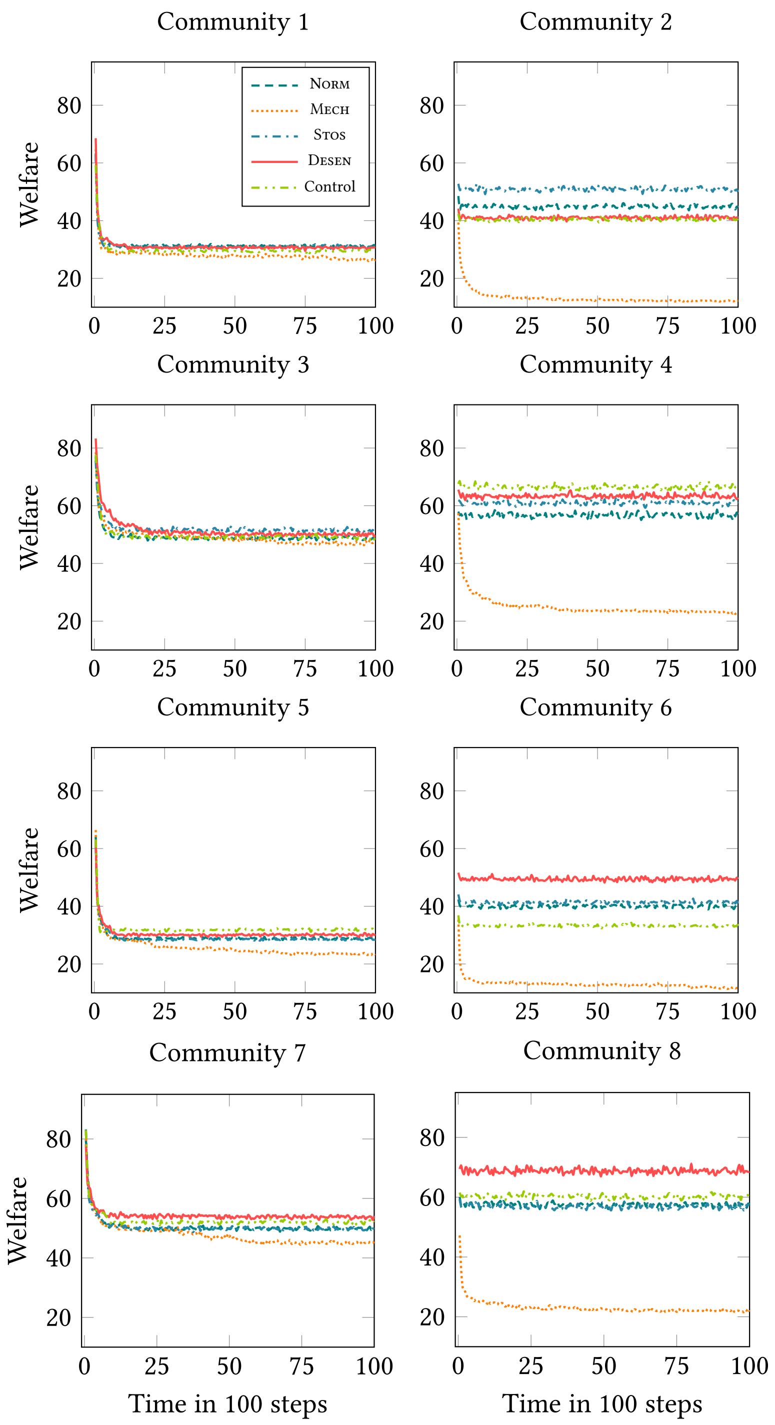
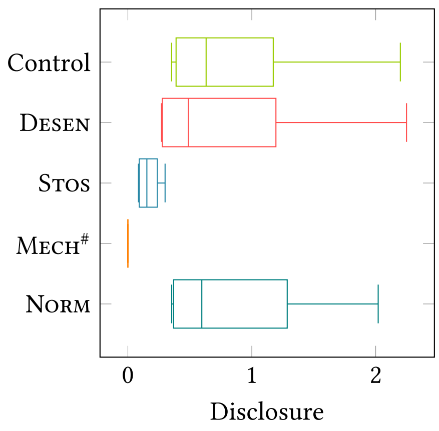
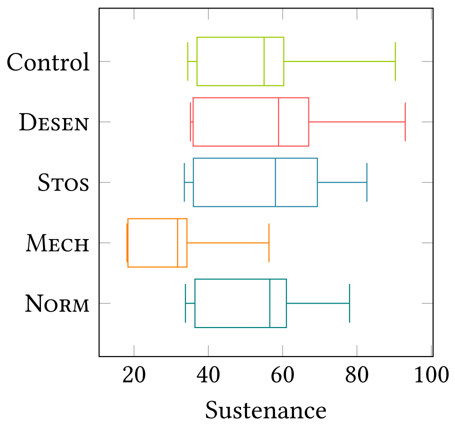


Fig. 11. Social welfare for the five solutions.



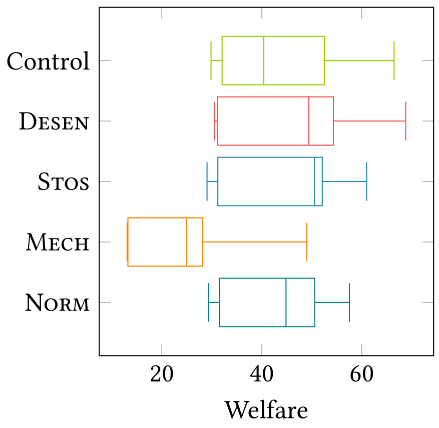
$\mu_{\text{Mech}}(0.0^{\#}) < \mu_{\text{Stos}}(0.19) < \mu_{\text{Desen}}(0.95) < \mu_{Control}(0.97) = \mu_{\text{Norm}}(0.97)$

Fig. 12. Boxplots comparing average information disclosure in the five solutions across eight community settings. # No disclosure in Mech.



$\mu_{\text{Desen}}(61.71) > \mu_{\text{Stos}}(59.15) > \mu_{Control}(57.63) > \mu_{\text{Norm}}(56.91) > \mu_{\text{Mech}}(34.38)$

Fig. 13. Boxplots comparing average life sustenance yielded by the five solutions across eight community settings.



$\mu_{\text{Desen}}(48.75) > \mu_{\text{Stos}}(46.65) > \mu_{Control}(45.55) > \mu_{\text{Norm}}(44.97) > \mu_{\text{Mech}}(28.11)$

Fig. 14. Boxplots comparing average social welfare yielded by the five solutions across eight community settings.