Data & Knowledge Engineering 87 (2013) 339–353

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Trust-based specification of sociotechnical systems

Elda Paja [a,*], Amit K. Chopra [b], Paolo Giorgini [a]

[a] Department of Information Engineering and Computer Science, University of Trento, Italy
[b] School of Computing and Communications, Lancaster University, United Kingdom

ARTICLE INFO

Article history:
Accepted 25 September 2012
Available online 23 December 2012

Keywords:
Conceptual modeling
Methodologies and tools
Sociotechnical systems
Roles
Trust

ABSTRACT

Current approaches in sociotechnical systems consider trust to be either *cognitive*—referring to actors' mental models of each other—or *technical*—referring to an actor's trust of a technical artifact. In this paper, we take a more expansive view of trust: in addition to the cognitive, we also consider trust in the architectural sense, which we term *sociotechnical trust*. Broadly, sociotechnical trust applies at the level of roles. Our principal claim is that sociotechnical systems are essentially specified in terms of sociotechnical trust. Whereas previous work has considered dependencies between actors as a fundamental social relation, we claim that no dependency can exist without the corresponding sociotechnical trust.

Our contributions are threefold. One, we qualitatively show sociotechnical trust to be different from cognitive and technical trust and the prevalent notions of dependencies among actors. Two, we specify a conceptual model of systems based on sociotechnical trust. We introduce the novel idea of one trust relation supporting another, which enables us to compare sociotechnical systems for trustworthiness. Three, we specify a methodology for engineering trustworthy sociotechnical systems. We evaluate our approach by modeling some aspects of a European food safety legislation.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

This paper considers the problem of modeling sociotechnical systems involving multiple autonomous actors such as organizations and humans. Autonomy means not only that none of the actors has control over other actors, but also that in general each actor would be independently motivated and would have independently-designed information systems that reflect its own motivations. This means that control-based abstractions for modeling information systems such as Web services, workflows, UML, and so on, are inadequate for modeling sociotechnical systems. We would instead need to model the interactions and the social relationships among the actors. Consider a modern health care system for example. It would involve hospitals, doctors, patients, laboratories, funding agencies, regulatory authorities, insurance companies, and so on, each of which is an autonomous actor. Consider, for instance, that no doctor controls any patient. However, they enter into meaningful social relationships with each other, for example, to write prescriptions, in the context of the health care system.

The importance of modeling the relationships among the actors is increasingly recognized by the software engineering and modeling community. Various kinds of relationships have been proposed in the literature. These include i*'s *intentional dependencies* [1], *commitments* [2], *norms* [3,4], and *responsibilities* [5]. In the software engineering community, the notion of intentional dependencies has proved particularly influential. We take the notion of dependency among actors as our point of departure.

* Corresponding author. Tel.: +39 329 88 234 35.
E-mail addresses: elda.paja@unitn.it (E. Paja), a.chopra1@lancaster.ac.uk (A.K. Chopra), paolo.giorgini@unitn.it (P. Giorgini).

0169-023X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2012.12.005

We acknowledge the importance of the notion of dependency among actors: any sociotechnical system exists in the first place because actors depend on each other to get things done. Differently, however, from the approaches in the literature (including i*), we propose capturing dependencies among actors in terms of *sociotechnical trust* relationships. It is essentially trust that makes dependence on others reasonable: when an actor trusts another for something, it expects the latter to do that thing. Lacking such a trust relationship, an actor can hardly depend on another for anything. Let's return to the health care setting. Let Alice and ModernLabs be a patient and a laboratory, respectively. Alice would be unlikely to make a payment to ModernLabs if she did not trust ModernLabs to deliver test results upon payment.

Our key intuition is that sociotechnical systems are organized along trust relationships. We describe an approach for modeling sociotechnical systems in terms of trust relationships. In contrast to a wide range of approaches on trust, it models trust at the level of roles in a sociotechnical system. Thus, for example, the above relationship between Alice and ModernLabs would in fact hold between every patient–laboratory pair participating in the health care system.

Further, we want to be able to compare two sociotechnical systems in a particular domain, and be able to objectively say which system fares better from the trust perspective. For example, all other things being equal, one could objectively say that the health care system which encodes the above trust relationship between patients and laboratories is better for (all) patients than the system without. "Objectively" means that anyone, even someone who has no intention or need of participating in the systems being compared, would come to the same conclusion. We make such comparisons all the time in our day to day lives. For example, we intuitively know that from a customer's point of view, an online marketplace that mandates that merchants refund customers for returned products within a month of purchase is better than one that does not allow such returns; the customers trust the merchants more in the former. It is the same reason why we deem credit card holders to be better off in the credit card system resulting from the passage of the Credit Card Act of 2009 [6]—in the new system, credit card holders now trust that banks will not arbitrarily raise interest rates, and so on. This paper explores the conceptual basis behind such intuitions.

1.1. Contributions

Our notion of sociotechnical trust derives from Singh's notion of architectural trust [7]. Whereas Singh elaborates on the formal semantics, our contributions lie in applying this notion to the engineering of sociotechnical systems, for which we coin the term *sociotechnical trust*. Our contributions are the following.

- We give an interpretation of the important notion of dependency among actors in terms of trust among actors. We show how this notion of dependency is different from those in the literature. We also show how the notion of sociotechnical trust is different from those of cognitive and technical trust, which are two kinds of trust relationships that have received the most attention in software engineering and computing.
- We characterize the notion of what it means for a sociotechnical system to engender more trust than another. We do this via the notion of one trust relationship *supporting* another trust relationship. We then apply this notion toward formulating a criterion by which one can claim if a system is more trustworthy than another from a particular role's perspective.
- We present a methodology for modeling sociotechnical systems based on the *supports* relationships. We evaluate our methodology against the European Food Safety Legislation (understood as a sociotechnical system for ensuring food safety).

As a matter of convention, the terms 'system' and 'trust' refer to sociotechnical system and sociotechnical trust, respectively, unless explicitly qualified otherwise.

1.2. Organization

The rest of the paper is organized as follows. Section 2 motivates and introduces the notion of sociotechnical trust. It also discusses how sociotechnical trust is different from the prevalent notions of trust. Section 3 introduces a conceptual model of sociotechnical systems. It also discusses the supports relation and defines what it means for one system to be more trustworthy than another. It also presents a number of trust-supporting patterns. Section 4 presents a methodology for specifying sociotechnical systems. Section 5 evaluates our methodology against a case study from the European Food Safety Legislation. Section 6 discusses literature related to the broad themes in this paper and concludes with a discussion of future work.

2. Sociotechnical trust

Our motivation behind proposing sociotechnical trust is to capture the essence of dependencies among actors. Currently, dependence is broadly understood in terms of the goals of the actors. The notion of intentional dependency in Yu's i* [1] has proved particularly influential. It sought to capture the idea that actors often depend on others for achieving their goals. Tropos [8] then went on to build a software engineering methodology around this idea of intentional dependency. Separately, in the multiagent systems community, Sichman et al. [9] introduced the notion of one actor depending on another for achieving its goals and give procedures for inferring this dependence.

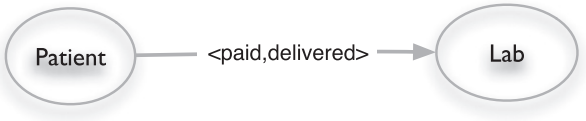

Fig. 1. The trust relation T_1.

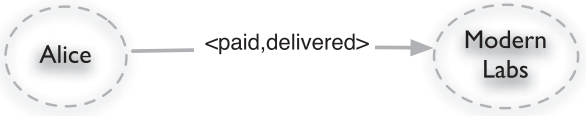

Fig. 2. Instantiation of the trust relation T_1.

The problem, however, with goal-oriented formulations of dependencies is that they refer to the internal goals and capabilities of actors. This means that the internal constructions (including the goals) of the actors involved would themselves be public. In sociotechnical systems involving autonomous actors, however, the actors would normally keep their internal constructions private. Further, actors can potentially join and leave the system at will. So it is not even clear whose goals the dependencies refer to. Consequently, any formulation of dependency based on actors' goals amounts to a brittle unwarranted assumption. What we need is a formulation of dependencies in terms of public abstractions.

With sociotechnical trust, we wish to capture the notion of dependence without referring to actors' internals. We use the notation $\mathrm{T}(x, y, p, q)$ to mean that x trusts y for q if p; here x and y are the truster and the trustee role, respectively, and p and q are the antecedent and the consequent, respectively. The antecedent and consequent are propositions specified in some formal language (in this paper, propositional logic). For example, $\mathrm{T}(\mathsf{Alice}, \mathsf{ModernLabs}, \mathsf{paid}, \mathsf{delivered})$ means that Alice trusts ModernLabs to deliver the results if she has paid for the tests; $\mathrm{T}(\mathsf{Alice}, \mathsf{ModernLabs}, \top, \mathsf{delivered})$ means that Alice unconditionally trusts ModernLabs for delivery. In contrast to goal-oriented formulations, the antecedent and consequent do no refer to the goals of the actors, but conditions of the environment that are in principle observable. For example, neither paid nor delivered refer to anyone's goals; they may refer to events corresponding to the occurrence of certain messages the involved actors exchange.

Because sociotechnical trust does not refer to actor internals, it can be applied between roles. For example, one can express that the role Patient depends on the role Lab for delivery of results upon payment, that is, T_1

$$\mathrm{T}(\mathit{Patient}, \mathit{Lab}, \mathit{paid}, \mathit{delivered}) \qquad (T_1)$$

Fig. 1 shows T_1 graphically: roles are represented by ovals and a trust relationship among two roles is represented by an arrow pointing from the truster toward the trustee labeled with the <*antecedent*, *consequent*> pair. When Alice and ModernLabs adopt Patient and Lab respectively, the relation T(Alice, Modern, Labs, paid, delivered) is *instantiated*. Fig. 2 shows the instantiated trust relation. The only difference from Fig. 1 is that actors are represented by dotted ovals.

The distinction between roles and actors and the relationship among them is crucial. Actors are not instances of roles; they *adopt* roles. As you will see in Section 3, sociotechnical systems are specified with reference to roles, not individual actors.

Dependence without sociotechnical trust would be meaningless. One can potentially imagine that patients depend on laboratories for delivery, but that they do not trust so; however, it is not clear what dependence means in this case.

The notion of trust is a complex one and there are many different readings of what it means. Next, we distinguish sociotechnical trust from two important kinds of trust relationships in the literature: *technical* and *cognitive*. We show with the help of examples that the three are orthogonal notions, and that each plays a role in the smooth functioning of a sociotechnical system.

2.1. Technical trust

By actors, we refer only to *social* entities. In practice, this means only humans and organizations (or their software surrogates). A hospital (an organization) may provide the service of appointment scheduling via a Web application; clearly, the application itself is not an actor in the same sense that a hospital is. Similarly, a laboratory may use several devices in providing testing services to patients, for example, a CT scanner; clearly, the CT scanner too is not an actor. Thus, in this paper, we do not talk about *technical trust*—whether the hospital trusts the scheduling Web application (to work well) or whether the laboratory trusts the CT scanner. Broadly, technical trust is a relation where at least one of the

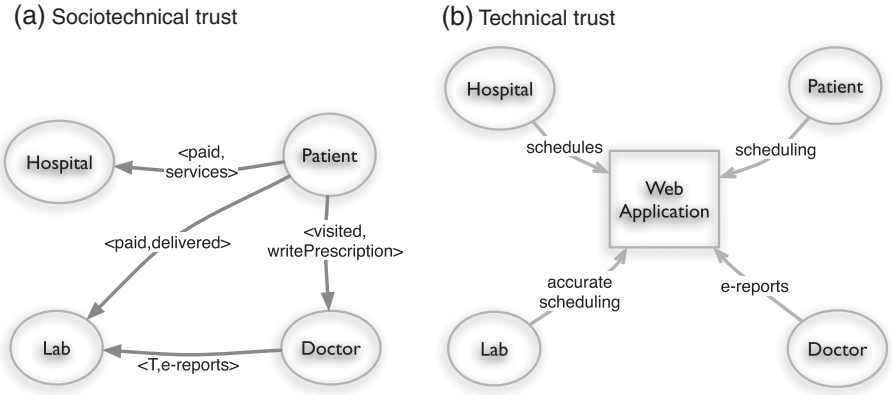

Fig. 3. Sociotechnical versus technical trust.

"actors" is a non-social component. We broadly identify technical trust with assurance [10], also sometimes referred to as dependability [11]. Many *trustworthy computing* initiatives including Microsoft's famed one [12] fall under the category of technical trust.

The contrast with technical trust is an important one. Technical trust is often applied in conceptually centralized settings where an information system can be deployed and evaluated for desirable properties. However, sociotechnical systems are not monolithic. Each actor is necessarily autonomous and implements the functionality it desires independently from other actors. For example, a hospital would implement its information systems independently from a laboratory. Further, they may have no visibility into each other's internal information systems. Therefore, whereas it may be prudent to dwell upon whether a hospital or a patient considers a laboratory's information systems and devices dependable, in practice it may turn out to be impractical to make any claims about them. However, it is critically important that the patient trusts the laboratory for providing accurate test results. Fig. 3 illustrates in an architectural sense the difference between sociotechnical and technical trust. As Fig. 1 shows, sociotechnical trust holds between social entities, such as Patient, Lab, Hospital, and Doctor. As Fig. 1 shows, technical trust involves technical non-social components, such as the Web Application. Incidentally, the traditional model of sociotechnical systems from requirements engineering is similar to Fig. 1.

2.2. Cognitive trust

Most of the predominant computational approaches to trust have a cognitive bias. In such approaches, each agent has a mental, and therefore necessarily private model of other agents based on which it chooses with whom to interact. Some approaches to trust are based on reputation. Although reputation itself is a social concept in that an agent's reputation is public, it also is mostly applied in a cognitive way—as an input to the agents' mental models.

As representative of the literature of cognitive trust, we consider Castelfranchi and Falcone's highly influential work on cognitive trust [13]. Castelfranchi and Falcone ask "what kind of beliefs and goals are necessary for trust?" and then go on to formulate various necessary conditions such as the truster having a goal g and the belief that the trustee is competent and willing to achieve g. Further, the truster should *rely* on the trustee to achieve g, meaning that the truster should not try to achieve g itself. In this manner, Castelfranchi and Falcone construct a rich picture of the mental state of actors. Let $\mathcal{T}_C$ denote the cognitive trust relation. For example, $\mathcal{T}_C(Alice, ModernLabs, paid, delivered)$ means that Alice cognitively trusts ModernLabs for delivery upon payment.

Cognitive trust is an important kind of trust, but just the same as intentional dependencies, it refers to the internal states of particular actors. Hence, it does not help us in the specification of sociotechnical systems involving autonomous actors. Nevertheless, there is an interesting connection between cognitive and sociotechnical trust. Sociotechnical trust modeled at the level of roles can inform an actor about the nature of the system, irrespective of particular actors that may participate in the system (we shall return to this point in more detail in Section 3 when we discuss how one system can be considered more trustworthy than another). Cognitive trust, by contrast, helps answer the question of whom to interact with. For example, let us suppose that there is a system where T_1 holds. Also suppose Alice adopts Patient, while ModernLabs and JohnsonLabs both adopt Lab in this system. Therefore, T_1 would be instantiated into both T(Alice, ModernLabs, paid, delivered) and T(Alice, JohnsonLabs, paid, delivered). However, cognitively, Alice trusts only ModernLabs, not JohnsonLabs. In other words, $\mathcal{T}_C(Alice, ModernLabs, paid, delivered)$ holds, but $\mathcal{T}_C(Alice, JohnsonLabs, paid, delivered)$ does not. In that case, it would seem prudent for Alice to choose to interact with ModernLabs rather than JohnsonLabs. In general, sociotechnical and cognitive trust are orthogonal: an actor may well choose to interact with a cognitively-trusted actor in the context of a system where

the appropriate sociotechnical trust relations do not hold rather than interact with a cognitively-untrusted actor in a system that has the appropriate sociotechnical trust relationships.

3. A conceptual model of sociotechnical systems

Following [14], we conceptualize a sociotechnical system as a specification of trust relationships with reference to roles, not particular actors. Let ϕ be a set of symbols. Let $\mathcal{P}$ be the set of all propositions over ϕ (including T, the constant for truth) using the connectives of propositional logic. Let p, q, … range over $\mathcal{P}$. Let R be a set of roles and let x, y, … be variables over roles and agents (whether they are over role or agents will be clear from the context of usage). We use T_1, T_2, … as labels to refer to trust relationships.

Definition 1. A sociotechnical system S is a set such that $S \subseteq \mathcal{R} \times \mathcal{R} \times \mathcal{P} \times \mathcal{P}$. We say T(x, y, p, q) if and only if $(x, y, p, q) \in S$.

For instance, consider a system HS1 in the health care domain with two roles Patient and Lab. In this system, the Patient trusts the Lab that it will provide accurate test results (T_2). HS1 is characterized as $\{T_2\}$.

Consider another health care system HS2 that in addition includes the role SpecializedLab, referring to a laboratory specialized for particular kinds of tests. In addition to T_2, the Lab trusts the SpecializedLab to analyze test outcomes and provide accurate results (T_3). Also, the Patient trusts the Lab that personal data will be confidential and not disclosed to other third parties (T_4). HS2 is characterized as $\{T_2, T_3, T_4\}$.

$$\mathrm{T}(\mathsf{Patient}, \mathsf{Lab}, \mathsf{takeTest}, \mathsf{receiveAccurateResults}) \quad (T_2)$$

$$\mathrm{T}(\mathsf{Lab}, \mathsf{SpecializedLab}, \top, \mathsf{provideAccurateResults}) \quad (T_3)$$

$$\mathrm{T}(\mathsf{Patient}, \mathsf{Lab}, \mathsf{providePersonalData}, \mathsf{ensureConfidentiality}) \quad (T_4)$$

3.1. Comparing sociotechnical systems

Consider T_1. To make this relationship more robust, one can think of other trust relationships, for example, that in the case of an excessive delay, patients trust laboratories to notify them about it (T_5), or that patients trust regulatory authorities to perform regular audits on the performance of laboratories to guarantee receiving accurate results (T_6), and so on. From the patient's point of view, both T_5 and T_6 *support* T_1.

$$\mathrm{T}(\mathsf{Patient}, \mathsf{Lab}, \mathsf{excessiveDelay}, \mathsf{notified}) \quad (T_5)$$

$$\mathrm{T}(\mathsf{Patient}, \mathsf{RegulatoryAuthority}, \mathsf{auditedLab}, \mathsf{receiveAccurateResults}) \quad (T_6)$$

We introduce the notation $\mathrm{T}(x, y, p, q) \succ \mathrm{T}(x, z, r, s)$ to mean that T(x, y, p, q) supports T(x, z, r, s) from x's perspective. Thus, for example, $T_5 \succ T_1$ and $T_6 \succ T_1$. Intuitively, a trust relation supports another if it potentially helps dealing with exceptions related to the latter.

Fig. 4 depicts that T_5 supports T_1. The *supports* relation is represented by an arrow pointing to the trust relation that is being supported.

One can imagine other kinds of relationships that potentially support T_1. For example, T_7 assures patients that their stored personal data will remain confidential; T_8 assures a patient that upon written consent, the results may be delivered to some other person named by the patient.

$$\mathrm{T}(\mathsf{Patient}, \mathsf{Lab}, \mathsf{personalDataStored}, \mathsf{confidentialityEnsured}) \quad (T_7)$$

$$\mathrm{T}(\mathsf{Patient}, \mathsf{Lab}, \mathsf{consentGiven}, \mathsf{deliveredToNamedPerson}) \quad (T_8)$$

We provide now a definition of what it means for a system to be more trustworthy from a role's perspective.

Definition 2. Let S_1 and S_2 be two systems. We say that S_2 is more trustworthy than S_1 from x's perspective ($S_2 \gg_x S_1$) if and only if:

1. $(x, y, p, q) \in S_1$ implies $(x, y, p, q) \in S_2$, and
2. $\exists (x, z, r, s) \in S_2$ such that $\mathrm{T}(x, z, r, s) \succ \mathrm{T}(x, y, p, q)$

Thus, for example, $\{T_1, T_6, T_5\} \gg_{Patient} \{T_1\}$.

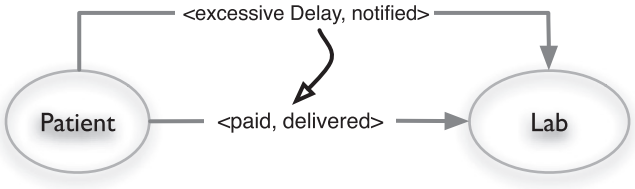

Fig. 4. Graphical representation of the supports relation.

Definition 2 gives potential participants and designers a basis for comparing sociotechnical systems. It captures the intuition that systems that make more provisions for exception handling are better. Of course, this does not mean that actors will adopt roles in systems that are more trustworthy in the above sense—they may have other requirements because of which they may favor participating in systems that are less trustworthy. Nonetheless, Definition 2 can serve as one among potentially many criteria for evaluating systems.

3.2. Patterns of trust

When specifying a system in terms of trust, the engineer would potentially try to identify the *supports* relations relevant to the system under consideration. Among the identified relationships, he or she would include in the system specification those that accord with the stakeholder requirements. Identifying the supports relations, though, is nontrivial. They are, generally speaking, domain-dependent. For any particular application, they are elaborated as a result of going through the domain description, which includes the regulations and rules governing the sociotechnical system. However, there are some recurring supports patterns that seem to apply across domains. Table 1 presents a number of supports patterns, which we call *patterns of trust*, and are induced from the set of patterns presented in [15,16].

Trust relationships among actors evolve with interaction [7]. In Table 1, the propositions *violated*, *delegated*, *threatened*, *undo*, *unreasonable*, *renegotiated*, and so on, represent potential states in the evolution of these relationships. We will show by means of examples how these *patterns of trust* help enhance trust in a system.

Referring to our health care running example, in case the Lab cannot deliver the test results to the Patient, then it can rely on a SpecialCourier to take care of that, offering yet another way to satisfy the Patient's needs. *Delegating* the responsibility of delivering the results to the SpecialCourier makes the relationship between Patient and Lab more robust. Recall T_1 (T(Patient, Lab, paid, delivered)). Let T_i be T(Patient, Lab, threatened(Patient, Lab, paid, delivered), delegated(Lab, SpecialCourier, delivered)); then $T_i \succ T_1$.

We represented *delegation* here as covering for potentially threatened trust relationships; however this pattern can be used for any situation in which one party relies on another to achieve a desired objective.

DelegationAcceptance is used to set up the proper relationship between a delegator and a delegatee for effective delegations. Generally, the delegator (Lab) may delegate to other roles (SpecialCourier), as shown in the first example, either when it cannot satisfy the trust relationship itself or when it is more convenient to rely on others. However, the delegatee is not bound to accept the delegation. DelegationAcceptance(Patient, Lab, SpecialCourier, paid, delivered) means that Patient will trust more a health care system in which the designated courier (in our example SpecialCourier) acknowledges and accepts the delegation from the laboratory, rather than the health care system in which such acceptance does not occur. The idea behind delegationAcceptance is that if the dependencies within the sociotechnical system are made stronger by being accepted, they can support the interactions of potential participants depending on the entities operating in the sociotechnical system.

MutualTrust is used to create reciprocal relations between the interacting parties. If one role trusts the other for u conditioned upon r, then the latter trusts the former for r conditioned upon u. This pattern aims to set up a richer and more flexible engagement, wherein both parties trust each other. MutualTrust(Patient, Lab, paid, delivered) states that not only Patient trusts the Lab to deliver results once paid, but also the Lab trusts Patient to pay for the tests upon delivery. This pattern is less prone to violations than when we have just a unilateral relationship, as only one party could possibly violate its trust relationship, and thus supports T_1.

Compensate and revert can be used to capture return–refund scenarios. Let T_j be T(Patient, Lab, violated(Patient, Lab, paid, delivered), refund ∧ discountCoupon). Then $T_j \succ T_1$. A trust relationship that ensures refunds along with a discount coupon for future

Table 1
Some patterns of trust.

Name	Trust Encoding
delegation(x, y, z, p, q)	T(*x*, *y*, *threatened*(x, y, p, q), *delegated*(y, z, p, q))
delegationAcceptance(x, y, z, p, q)	$T(x, y, T(y, z, p, q) \wedge delegation(x, y, z, p, q), accepted(z, y, p, q))$
mutualTrust(x, y, r, u)	T(x, y, T(y, x, u, r), T(x, y, r, u))
compensate(x, y, p, q, r, s)	$T(x, y, violated(x, y, p, q), T(x, y, r, s))$
revert(x, y, p, q, r)	T(x, y, *revert*(q), T(x, y, r, *revert*(p)))
renegotiate(x, y, p, q, r, s)	$T(x, y, unreasonable(p, q), T(x, y, renegotiated(r, s), T(y, x, r, s)))$
separationOfDuty(x, y, z_1, z_2, p, q_1, q_2)	T(x, y, p, *sod*(z_1, z_2, q_1, q_2))

tests makes the relationship more robust from the patient's point of view: the laboratory would compensate for the violation by offering a discount coupon and undoing the payment done by the patient. Thus, the application of this patterns would support the patient's initial trust in the laboratory for delivering test results. As a consequence, we say that the patient considers the latter system to be a better choice for him (more trustworthy).

Renegotiate, on the other hand, offers more alternatives for the role to choose. For example, in case of T_1 the laboratory might deliver the results of the test in a moment of time that is considered late for the patient. The patient may renegotiate for the time of the delivery to have the laboratory deliver the results ahead of time, by perhaps making a special request. Let T_k be T(Patient, Lab, paid, delivered, specialRequest, deliverAheadTime), as a result of applying renegotiate. Renegotiating other properties other than the initial one concerning the delivery of results, makes the interaction among the patient and laboratory more complete and robust, therefore the application of this pattern would lead to a more trustworthy health care system from the patient's point of view. So, we would have $T_k \succ T_1$.

SeparationOfDuty ensures more reliable interactions. For instance, if T_l stands for T(Patient, Lab, Doctor_1, Doctor_2, takeTest, performTest, approveResults) representing the pattern separationOfDuty, it supports the Patient's trust in the Lab to receive accurate results after taking the test (T_2). It does so by ensuring that the same doctor does not perform the test and approve the results for the same test, that is, the roles Doctor_1 and Doctor_2 must not be adopted by the same doctor actor. This means that if doctor Doctor_1 is performing the test to the patient, then he or she cannot also approve the results, Doctor_2 should do it instead. This double checking of results supports thetrust relationship the patient has toward the lab for delivering accurate test results, in other words $T_l \succ T_2$.

Two points are worth emphasizing here. One, the intuition behind introducing the supports relation is exception-handling. If there is a supporting trust relation for a particular trust relation, then exceptions pertaining to the latter can potentially be handled within the system itself. In the absence of a supporting trust relation, the exception would have to be dealt with outside the system. Two, deciding whether a trust relationship supports another is not a trivial task. For example, one could argue that $T_i \not\succ T_1$ on grounds that a patient may prefer not to have delivery outsourced to a third party. Our assumption is that domain experts would identify the supports relationships for the application under consideration.

4. System specification methodology

Our conceptual model of sociotechnical systems binds together roles via trust relationships. Now, we describe a methodology for coming up with the specification of a system in terms of trust relationships starting from a set of stakeholder requirements. Besides the requirements themselves, the methodology relies on domain knowledge in order to come up with the trust supporting relationships. The methodology is iterative: one comes up with a specification from requirements, identifies and then fills the potential gaps via the domain-specific supporting relationships and the patterns of trust, and then accounts for changes in requirements.

I. Initial Specification

Step 1. Identify the roles

Begin by gathering a description of the application and its general domain. This includes the requirements. Then identify the roles that participants can potentially adopt in the system.

Generally, in a domain description, there is information about who does what. For instance, in a service-oriented setting, the operating roles would be user and provider of the requested service. Some of the requirements would amount to interactor expectations, which can be used to identify the roles in the system. Earlier, we listed the participants in the health care example.

Step 2. Analyze social interactions

i. *Identify interactions*: After identifying the participating roles, identify the interactions they are involved in. For instance, if we want to specify the system in terms of trust relationships from Patient's perspective, then we need to identify the interactions it participates in. If Patient has the requirement to take some tests at the Lab, it might have several interactions with the Lab to achieve this requirement. First, the Patient may take an appointment, then take the test, and finally make the payment to receive test results. All these are interactions that take place between Patient and Lab in order for Patient to achieve its requirement of taking tests at the Lab. Similarly, for other roles we identify the interactions they are involved in. These are not only bound to specific requirements, but can hold for different purposes.

ii. *Group interactions and identify trust relationships*: For each role and for each requirement, group the various interactions that take place. As shown from the example in the previous step, for a certain requirement there can be several interactions between a pair of roles. Therefore, we can group these interactions and abstract from them *trust relationships*. We can identify either several trust relationships related to one requirement of a particular role or a representative trust relationship for the whole interaction. For simplicity, we choose to have one trust relationship per requirement. In our example, we consider the trust specification as containing the trust relationship between Patient and Lab, in which Patient trusts the Lab to have the test results *delivered*: T(Patient, Lab, paid, delivered).

Step 3. Identify the supports relations

After having identified the basic trust relationships, look for ways for making the system more robust to exceptions; that is, identify the supporting trust relationships. Look for domain-specific information that offer mechanisms to support the trust relationships. Starting from a role-based perspective, we consider the mechanisms that support the trust relationships this role has, thereby supporting its given requirements. For instance, in our running example, we want to come up with a trust specification that makes a given health care system more trustworthy from the Patient's

perspective. For this, starting from one requirement the Patient has (*receive accurate test results*), we have a first specification of trust in terms of the trust relationship established between Patient and Lab. But, we want to ensure this trust relationship holds, that is why we look for mechanisms that support this trust relationship, and as a result support the achievement of the Patient's requirement. Suppose that the Lab uses a particular device to analyze samples taken from the Patient. *Checking the device periodically* (say every night) *supports* receiving accurate results. From this, we can abstract a *supports* relation, namely: *periodical checks*.

In order to identify the supports relations, we look at the *domain description* obtained in the first step and consider domain-specific information. For instance, in the health care example, we consider information about how particular procedures are performed and what happens if the normal flow is not followed. Potential sources for identifying the supports relations include contracts (especially the clauses related to authorizations, obligations, prohibitions or penalties), manuals, procedural regulations, and so on.

II. Enhancing trust: Making the initial system specification more robust

Step 1. Apply the supports relations

For the given requirements, if we discover trust supporting relations for the trust relationships that represent these requirements, we apply them to obtain a more trustworthy system. Going back to our running example, we apply the trust relation *periodical checks* to support *receive accurate test results*.

Step 2. Identify gaps and apply patterns of trust

After applying the supports relations, we check again the specification of the system, and identify unhandled situations, that is, find the trust relationships that have no applicable supports relations. Further, we consider also the situations in which the applicable supports relations might themselves be threatened by exceptions. We then evaluate the possibility of handling the exceptions with the help of the *patterns of trust*.

We identify the applicable patterns and introduce them on top of the trust relations they support. For instance, if the Patient is not delivered the test results, we can apply the Compensate pattern, which refunds the Patient for the performed payment and provides him or her with a discount for the next time he or she needs to retake the test. This pattern makes the trust relationship between Patient and Lab more robust as it considers handling a situation in which things go wrong and offers a solution to it.

III. Revise: Check for new requirements

The process is iterative and it continues till it provides a system specification that supports the satisfaction of all the given requirements. However, requirements may change with time; so this is another trigger to go through the entire process again, this time taking into consideration the new requirements. These new requirements may bring up the application of new *supports* relations or *patterns of trust*. For instance, if patients have the requirement to have test results delivered from the laboratories, they might have an additional requirement of keeping track of the process from the moment of taking the test to the moment of obtaining the results. Traceability is another mechanism we can apply, to help patients satisfy their requirement, and provide a system specification that apart from ensuring the delivery of the test results, also ensures keeping track of the status of the whole delivery process.

Fig. 5 represents graphically the processes that take place in designing a trustworthy sociotechnical system. It shows the different steps our approach follows starting from the identification of an initial system specification (I.2), to the improvement of the system specification through *supports* relations (identified in I.3 and applied in II.1) and *patterns of trust* (II.2). The approach is iterative as new requirements and domain-specific information might emerge and trust relationships might be violated. We follow the steps of this process in a real case study regarding European Food Safety Regulation in Section 5.

5. Case study: the food safety regulation

The European Parliament and the Council have adopted Regulation (EC)178/2002 [17] to harmonize all Member States food legislation in a general Food Safety Regulation, whose primary objective is consumer protection throughout Europe. The regulation lays down all requirements on food safety, establishing the rights of consumers to safe food and accurate information. It imposes requirements on any substance that is intended or expected to be incorporated into a food (feed) during its manufacture, preparation or treatment [17]. These requirements must be applied by all food operators in order to comply with the regulation. Whenever a breach to a particular legal limit is identified, it might be the case that the product in question might be injurious to human health or unfit for human consumption [17]. Therefore, assessments need to be carried out to verify the product conditions. A crucial point of (EC)178/2002 regulation is traceability of food, whose aim is assuring that the origins of food or feed ingredients and food sources can be identified along the whole food (feed) chain from production to consumption. This is especially relevant when products are found to be faulty. It works on the basis of "one step back–one step forward" principle: every food operator should be able to identify the immediate supplier and the immediate recipient of the product at hand. For this reason, traceability is established at all stages of food (feed) chain. According to regulation (EC)178/2002, the responsibility for ensuring food safety lies with food business operators. Competent authorities perform controls to ensure compliance with the regulation.

In case products are not in compliance with food safety requirements or there are reasons to believe that this might be the case, food business operators should start the procedures for withdrawing these products from the market. As part of these procedures stands also the obligation to notify competent authorities. If the product at hand is considered to be unsafe or unfit for human consumption and it has already been released to the market, food operators should inform consumers accurately and effectively for

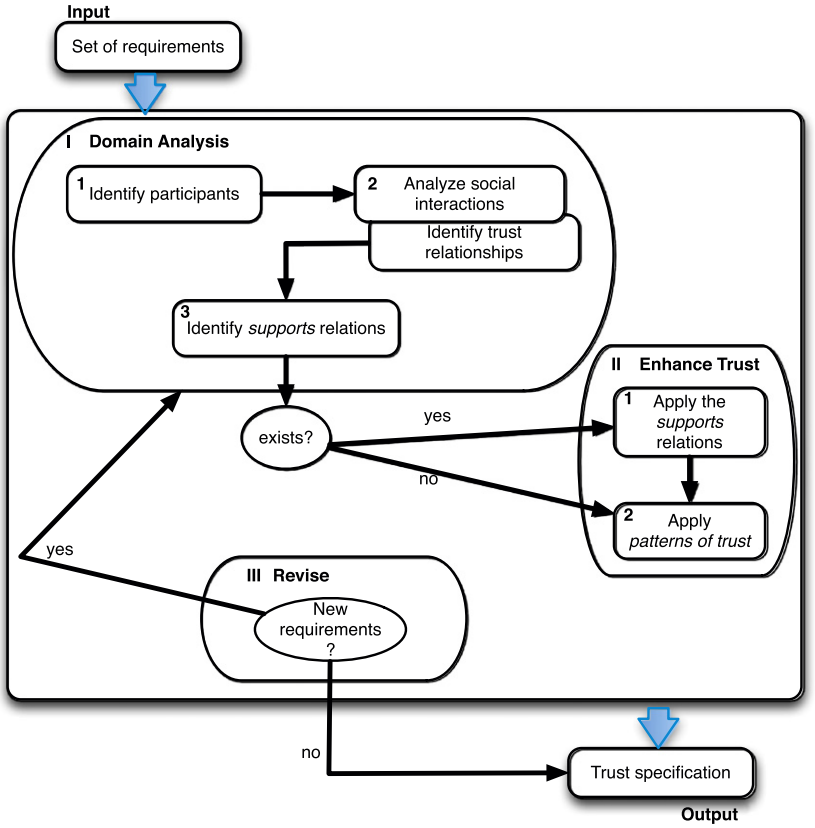

Fig. 5. Sociotechnical system specification methodology.

the reason of withdrawal and if necessary recall from them the products already supplied to them [17]. Recall means that the consumers either should return the product back where they purchased it or they should destroy it. Food business operators are responsible for taking these measures and collaborating with authorities, providing relevant information to them, and participating in traceability so that the end consumers face no risks. When it comes to import and export of food and feed, the regulation has no effect outside EU. As far as imports are concerned, requirements are imposed to the EU importer, who should be able to identify the source in the third country from whom the products were acquired. As for exports, EU regulations are in place, unless the third country specifies other requests, in which case these take precedence.

In the Food Safety Regulation case study, the core requirement is that consumers are provided with safe food, which means that every food placed on the market must be safe. Therefore, we will follow the steps of our iterative methodological approach to build the trust specification in order to enhance consumers' trust about the overall food chain.

5.1. Trust specification

We start from the requirement of *providing safe food in the market* and build the trust specification for the system that should satisfy this requirement.

I. Initial specification

The description of the domain includes the way the law imposes restrictions on the way food chain operates. We have already provided the description of the domain for the Food Safety Regulation at the beginning of this section.

Step 1. Identifying roles

First, we need to identify the roles present in the system, so that agents interested in participating can adopt the appropriate roles. The Food Safety Regulation exposes the following participating roles: Member States (MS), Food Safety Authority (FSA), Food Business Operator (FBO), European Consumer (EC), Third Country (TC). Notably, various stakeholders such as food (feed) manufacturers, importers, brokers, farmers, distributors are all classified as FBO.

Step 2. Analyze social interactions

Authorities (FSA) impose requirements over the FBO with regard to assessing the quality of the products they place on the market, providing accurate information to consumers, taking into account sensitivities of particular consumer groups, and categorizing food accordingly. The FSA holds the FBO responsible for the activities under their control.

European Consumers (EC) expect to have no harm by the food they buy and to have no injurious food placed in the market.

To be in compliance with (EC)178/2002 requirements and specifications, the trust relationships in Table 2 must hold. We include only the relationships from the consumers and food business operators perspective within the EU, not the ones dealing with import or export operations. We provide a semi-formal graphical representation in a multilayer perspective (Fig. 6) starting from the basic situation (container denoted with R) that represents the system in which EC (European Consumer) trusts FBO to place safe food on the market. This trust relationship is established to represent the core requirement that consumers are provided with safe food, that is, every food placed on the market must be safe. Food operators are responsible for ensuring this, therefore we show the trust relation that should hold between EC and FBO (T_9).

Step 3. Identify the supports relations

The regulation specifies mechanisms that can help improve this situation, thereby enhancing the chances of achieving the primary requirement of having safe food on the market. From the domain description, we can spot out mechanisms that deal with measures that need to be taken along the food chain, such as the obligations to notify authorities, to monitor, and to withdraw and recall products injurious to human health, and so on. These mechanisms are actually the *supports* relations specific to this domain. We identify: Monitor, Notify, Prohibit, Traceability, Withdraw or Recall, and Assess Risks.

II. Enhancing consumers' trust

Step 1. Apply the supports relations

We now apply the identified *supports* relations to enhance European consumers' trust in the system, to ensure they are provided with safe food. The structuring of the trust relationships in Fig. 6 serves the purpose of considering the different trust relationships that are established between EC and FBO along the different phases of the food chain. Thus, after specifying the first trust relationship (T_9) in the first layer, in the second layer we consider monitoring (Monitor: T_{14}) and notification (Notify: T_{15}) mechanisms. They consist in monitoring suppliers and food, and in notifying when food safety is at risk. As such, these two trust relationships support the initial trust relationship T_9. Monitoring increases the level of trust consumers have on the products placed on the market, given that a food safety authority is involved in the process (FSA). However, monitoring alone is not enough. Notification helps identify and handle risky situations, and on top of it *prohibition* (Prohibit: T_{10}) avoids placing unsafe food in the market. Thus, it further supports T_9.

Prohibiting unsafe food in the market is one special case, in which food is identified to be unsafe for human consumption even before being distributed. But, problems might be identified when this initial stage is already passed. That is why Traceability mechanisms (T_{13}) are applied on top of prohibition, monitoring and notification, so that any time a given risk is identified, it is possible to find the node in the food chain in which the breach occurred.

Table 2
Trust relationships based on role perspectives.

Trust relationships
European Consumers' Perspective
T_9. T(EC, FBO, onMarket(f), safe(f)) Consumers trust food business operators that every product they placed in the market is safe.
T_{10}. T(EC, FSA, violated(T_9), prohibitPlacingOnMarket(f)) Consumers trust the authorities that if food is found to be unsafe, it will be prohibited to be placed in the market.
T_{11}. T(EC, FBO, violated(T_9), T(EC, FBO, informOn(f), withdraw(f))) Consumers trust food business operators that if unsafe food is found on the market, they will be informed and the food will be withdrawn from the market.
T_{12}. T(EC, FBO, hazardsIdentifiedOn(f), assessRisksRelatedTo(f)) Food business operators should perform risk assessment and analysis for any potential hazard related to a given food product.
T_{13}. T(EC, FBO, product(f, FBO) ∧ ingredients(f, i_0) ∧ ... ∧ ingredients(f, i_n), record(i_0, $supplier_0$) ∧ ... ∧ record(i_n, $supplier_n$) ∧ record(f, customer)) Food business operators should keep record of all the suppliers of ingredients of the products they sell and of all the immediate customers.
Food Business Operators' Perspective
T_{14}. T(FSA, FBO, monitoredSuppliers, risksDetectedEarly) Problems with food will be detected early if food business operators monitor their suppliers.
T_{15}. T(FSA, FBO, risked(T_9), notify) Food safety authority expects to receive notification on risk.

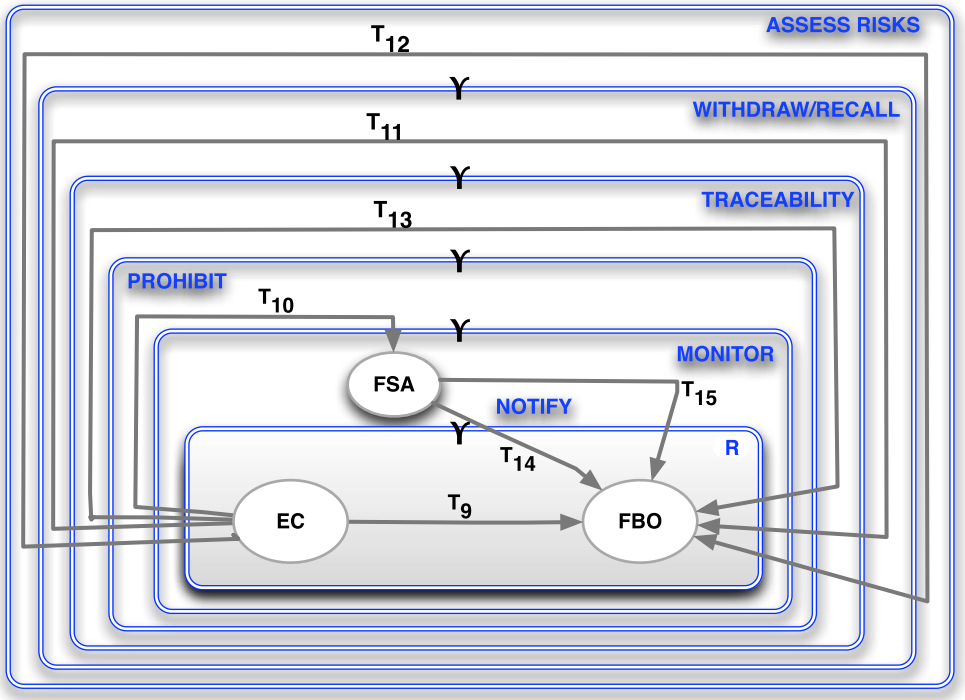

Fig. 6. Enhancing European Consumers' Trust: Applying *supports* relations.

Whenever a hazard is encountered, traceability is used to identify the source of such risk.
Food operators should perform risk assessment to verify the status of the given product. If food is found to violate safety requirements, consumers must be notified and the product needs to be withdrawn or recalled (Withdraw/Recall: T_{11}). Risk assessment procedures influence the decision to withdraw or recall a product from the market (Assess Risks: T_{12}). Notably, *assess risks* supports all the trust relationships introduced in the layers below.
For simplicity and readability of the figure, we do not use the notation introduced in Fig. 4 in Section 1, but instead we use the *supports* relationship symbol at the border between the subsequent layers, to represent the fact that trust relationships in the above layer *support* trust relationships in the layers below, enhancing at each subsequent layer consumers' trust about the system. As such, this structuring conveys the idea of how consumers' trust is increased by the application of supports relations.

Step 2. Apply patterns of trust
After applying the *supports* relations, we consider possible exceptions that can occur. Notice that in order to ensure that suppliers comply with the food safety regulation by providing food that is within the specified parameters, we used *monitoring* to support T_9. However, this supports relation would not work out if the actor that adopts FSA (the monitoring authority) adopts FBO too. To avoid this problem, we apply the pattern *SeparationOfDuty* (SoD, see Fig. 7). We represent the patterns as labels positioned right in front of the *supports* relations, whose possible exceptions they can handle. The same pattern can be applied over Assess Risks as well.
Moreover, food business operators have to carry out risk assessment procedures, but they might not have the capability or the right equipment to perform such procedures. Therefore, we apply the Delegation pattern to allow FBO to delegate this task to a more specialized entity. For simplicity, we do not introduce the new roles that emerge, but only the application of the *patterns of trust*.
In case a recall happens, and the consumers have to bring back the product, we apply the Compensate pattern to refund consumers for the product they have purchased, as well as to offer them a discount coupon for the following purchases. In this way, via the application of supports relations and patterns of trust, we handle the various exceptions that might arise in the trust relationship among the roles EC and FBO.

III. Revise
We do not consider here the introduction of new requirements in the food safety regulation case study.

5.2. Comparison

We presented a graphical representation of a sociotechnical system from the food safety regulation case study showing trust relationships that hold between the different roles (mainly consumers and food business operators) along the food chain. We used the

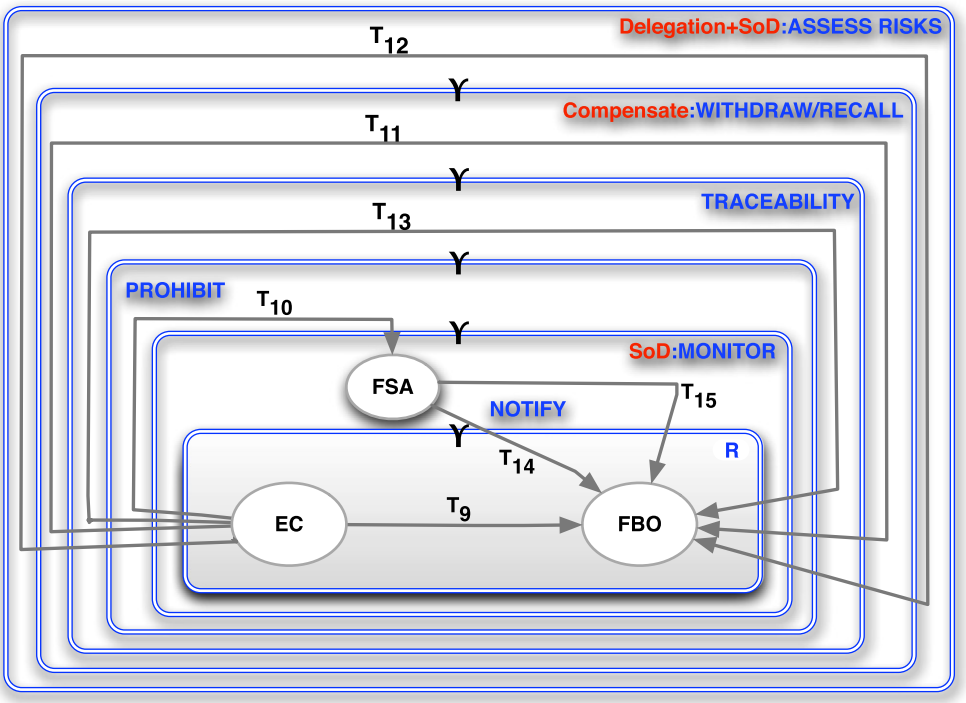

Fig. 7. Enhancing European Consumers' Trust: Applying *patterns of trust.*

case study to identify domain-specific supports relations that aim to increase the consumers' trust in this sociotechnical system. Based on the identified *supports* relations, we built a structured graphical representation of the trust relationships that hold in the system to help possible participants decide which system they want to play a role in. This representation makes the role perspective clearer by showing the relations they might be involved in. Furthermore, this structuring allows to see how trust is increased in a layered fashion by representing the trust supporting mechanisms and patterns of trust. All mechanisms represented in terms of *supports* relationships and the applied *trust patterns*, serve the purpose of ensuring consumers' safety and inspiring their trust in the system.

Is the approach scalable for larger case studies? Our approach depends on the clarity of the information regarding the considered domain, as the process of discovering *supports* relations is domain-specific. Once this step is carried out, the application of the *patterns of trust* is easier as it tries to fill in the gaps left by the *supports* relations or to handle possible exceptions that might arise from them.

6. Discussion

In this paper, we proposed sociotechnical trust as the essential meaning of the notion of dependency among actors in sociotechnical systems. We model a system in terms of trust relationships among roles in the system; when specific actors adopt the roles, those relationships are instantiated. Making the trust relationships explicit helps prospective participants decide what they can expect from other participants of the system and what is expected of them at a social level. We then advanced the idea of certain trust relationships supporting certain others, and used this notion to help compare sociotechnical systems from a role-based perspective. Thus, whereas a health care sociotechnical system may be more trustworthy from the patient's point of view, it may be less trustworthy from a laboratory's point of view. Such a perspective follows naturally from the fact that trust is a directed relationship.

We analyzed a European food safety regulation on the basis of our approach. The case study reveals that a sociotechnical system is not a list of trust relationships, but there is an underlying structure in the sense that some trust relationships enhance some others. Understanding the structure of a system will prove valuable for those who want to decide their participation in a system.

It is worth emphasizing that our notion of a system being more trustworthy than another relies upon domain experts and stakeholders identifying the supporting relationships, which may or may not agree with potential participants in the system. For a particular actor, a system that is more trustworthy by Definition 2 may be less suitable. For example, even though $\{T_1,T_6, T_5\} \gg_{Patient} \{T_1\}$, Alice may nonetheless prefer $\{T_1\}$ for whatever (potentially unknown) reason. This does not indicate a flaw in our definition; it highlights the fact that our notion of trust does not rely upon actor internals (here, Alice's intentions).

6.1. Sociotechnical systems

The field of sociotechnical systems emerged in the 1960s. Its principal concern was the *design of work* in organizations, meaning how to design organizational systems that took into concern not just the technical requirements of stakeholders, but also the social and humanistic concerns of users and employees [18]. Fields such as human–computer interaction (HCI) and computer-supported collaborative work (CSCW) have traditionally been at the forefront of sociotechnical systems research (for example, see [19] and [20]). Baxter and Sommerville [21] recommend a range of other fields and methods, (for instance, ethnographic studies), that software engineering needs to consider in order to move from the realm of the technical to the social.

The notion of understanding systems in terms of trust relationships among its actors sheds new light on the social structures of sociotechnical systems. There is one important difference though between the traditional conception of sociotechnical systems and our conception. The traditional conception emphasizes an organizational setting (with increasingly decentralized decision-making, as Mumford [18] suggests) whereas our conception emphasizes autonomy and logically-decentralized settings.

6.2. Software engineering

Baxter and Sommerville [21] conceptualize a sociotechnical system as a conceptually centralized system (potentially with physically distributed components) that pursues its goals and adapts accordingly. In our conception, the system does not have goals; it is a specification of trust relationships. This enables us to talk sociotechnical systems spanning multiple autonomous actors, each of whose goals are potentially unknown.

Requirements engineering modeling languages and methodologies such as i* and Tropos, although multiactor, also assume a single organizational setting, and thus are conceptually centralized as well [22]. Some researchers have forwarded the notion of *responsibility* of actors, either as requirements themselves or as their potential source. Dardenne*et al.*'s [23] notion of responsibility is a single-actor relation, not multiactor. Other work [5,24] advances responsibility as a multiactor relation; however, it is understood in terms of the obligations of actors. By contrast, trust implies no obligation; it is therefore a lighter, more elementary relation for understanding the structure of sociotechnical systems at a high-level.

Just the same as responsibilities though, the specification of systems in terms of trust relationships can serve as requirements for further engineering. As a normative relation, trust is somewhat weak—it is not enforceable. The specification can be used to come up with the specification of the commitments [25] (understood as elements of contracts) among actors. The advantage with commitments is that they are enforceable. Alice may trust ModernLabs for delivery upon payment, but in practice that may not be enough for her to act; she may also need the corresponding commitment from the laboratory. Trust and commitment seem to be orthogonal social notions. Neither implies the other. An actor can act solely upon the basis of trust or solely upon the basis of commitments, but the ideal situation would involve both trust and commitment. An interesting direction to explore would be the interplay between trust and commitment. Singh [26] echoes similar intuitions.

Checkland [27] contrasts the narrow idea of engineering as building systems given some objectives with the broader idea of *systems thinking*, which involves exploring a problem space that is both rich and confusing because of the complexity of human affairs. According to him, in situations dealing with human affairs, even stated objectives are never straightforward. Goguen [28] expresses a similar intuition about requirements when he claims that given the complexity of social affairs, requirements are potentially known only in hindsight. Like Baxter and Sommerville, both Checkland and Goguen point to the richness and difficulty of building systems to support human affairs. This bears upon the work we presented here: although we have a model of sociotechnical systems in terms of trust and a simple methodology for specifying such systems, we abstract away in the methodology from the details and richness of human affairs that would bear upon the modeling.

6.3. Trust

Castelfranchi [13] lists different kinds of trust that often come into play: trust in the environment and infrastructure (technical trust), trust in one's own agent (technical trust because it amounts to trust in the software that an actor uses), and trust in authorities and partners (both cognitive). These are all important kinds of trust; however none of these is sociotechnical. Asnar et al. [29] model trust relationships among actors in order to analyze risk; however in their approach, trust is an agent's subjective belief about another, in other words, cognitive. Similar to our approach, Giorgini et al. [30] model social trust at the role level. They present examples where any agent adopting some role must trust another adopting another role for something; however, cognitively it may not. Giorgini et al. deem this a conflict. In our approach, such a situation is not a conflict—social and cognitive trust are orthogonal concepts. Our idea of assuming domain-specific trust enhancing (the *supports*) relationships is not unfounded. Jones et al. [31] present a list of trust requirements for e-business. Haley et al. [32] introduce the notion of trust assumptions to help discharge concerns about system security.

6.3.1. Future directions

This paper is an elaboration of our intuitions that relate system specification to trust; therein, lies its novelty. Potentially interesting future directions include (1) considering the trust specifications themselves as requirements for engineering contracts among parties, (2) giving a richer account of the various social factors (as advocated in the sociotechnical systems literature) that go into system specification, and (3) making the intuition behind the supports relation more concrete by defining it in terms of exception handling.

Acknowledgments

The research was partially funded by the European Union Seventh Framework Programme (FP7/2007–2013) under grants no. 257930 (Aniketos) and 256980 (NESSoS). Amit Chopra was supported by a Marie Curie TrentinoCofund grant and the ERC Advanced Grant 267856 *Lucretius: Foundations for Software Evolution*. We are grateful to Munindar Singh for helpful discussions. The comments of the anonymous reviewers also helped improve this paper substantially.

References

[1] Eric Yu, Towards modelling and reasoning support for early-phase requirements engineering, in: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997, pp. 226–235.
[2] Nirmit Desai, Amit K. Chopra, Munindar P. Singh, Amoeba: A methodology for modeling and evolution of cross-organizational business processes, ACM Transactions on Software Engineering and Methodology (TOSEM) 19 (2) (2009) 1–45.
[3] Alberto Siena, Giampaolo Armellin, Gianluca Mameli, John Mylopoulos, Anna Perini, Angelo Susi, Establishing regulatory compliance for information system requirements: An experience report from the health care domain, in: Proceedings of the 29th International Conference on Conceptual Modeling, LNCS, volume 6412, Springer, 2010, pp. 90–103.
[4] Munindar P. Singh, Norms as a basis for governing sociotechnical systems, ACM Transactions on Intelligent Systems and Technology (TIST), 2013, pp. 1–30. (To appear); available at http://www.csc.ncsu.edu/faculty/mpsingh/papers.
[5] Ros Strens, John Dobson, How responsibility modelling leads to security requirements, in: Proceedings of the New Security Paradigms Workshop, 1993, pp. 143–149.
[6] Credit card accountability responsibility and disclosure act of 2009, http://www.govtrack.us/congress/bill.xpd?bill=h111-627.
[7] Munindar P. Singh, Trust as dependence: A logical approach, in: Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2011, pp. 863–870.
[8] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents and Multi-Agent Systems 8 (3) (2004) 203–236.
[9] Jaime SimãoSichman, Rosaria Conte, Yves Demazeau, Cristiano Castelfranchi, A social reasoning mechanism based on dependence networks, in: Proceedings of the 11th European Conference on Artificial Intelligence, 1994, pp. 188–192.
[10] Nancy R. Mead, Joe Jarzombek, Advancing software assurance with public–private collaboration, IEEE Computer 43 (9) (2010) 21–30.
[11] Guy Dewsbury, Ian Sommerville, Karen Clarke, Mark Rouncefield, A dependability model for domestic systems, in: Computer Safety, Reliability, and Security, LNCS, volume 2788, Springer, 2003, pp. 103–115.
[12] Steve Lipner, The trustworthy computing security development lifecycle, in: Proceedings of the 20th Annual Computer Security Applications Conference, dec 2004, pp. 2–13.
[13] Cristiano Castelfranchi, Yao-Hua Tan, The role of trust and deception in virtual societies, International Journal of Electronic Commerce 6 (3) (2002) 55–70.
[14] Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, John Mylopoulos, Modeling and reasoning about service-oriented applications via goals and commitments, in: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE), LNCS, volume 6051, Springer, 2010, pp. 113–128.
[15] Munindar P. Singh, Amit K. Chopra, Nirmit Desai, Commitment-based service-oriented architecture, IEEE Computer 42 (11) (2009) 72–79.
[16] Amit K. Chopra, Munindar P. Singh, Specifying and applying commitment-based business patterns, in: Proceedings of the Tenth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2011), 2011, pp. 475–482.
[17] Standing Committee on the Food Chain and Animal Health. Food law implementation guidelines.World Wide Web electronic publication, January 2010. Last checked: July, 2012.
[18] Enid Mumford, A socio-technical approach to systems design, Requirements Engineering 5 (2) (2000) 125–133.
[19] Fernando Flores, Michael Graves, Brad Hartfield, Terry Winograd, Computer systems and the design of organizational interaction, ACM Transactions on Information Systems 6 (1988) 153–172.
[20] Alistair G. Sutcliffe, Sarah Thew, Paul Jarvis, Experience with user-centered requirements engineering, Requirements Engineering 16 (4) (2011) 267–280.
[21] Gordon Baxter, Ian Sommerville, Socio-technical systems: From design methods to systems engineering, Interacting with Computers 23 (1) (2011) 4–17.
[22] Amit K. Chopra, Paolo Giorgini, Requirements engineering for social applications, in: Proceedings of the 5th International i* Workshop, *CEUR Workshop Proceedings*.CEUR-WS.org, volume 766, 2011, pp. 138–143.
[23] Anne Dardenne, Axel Van Lamsweerde, Stephen Fickas, Goal-directed requirements acquisition, Science of Computer Programming 20 (1–2) (1993) 3–50.
[24] Ian Sommerville, Russell Lock, Tim Storer, John Dobson, Deriving information requirements from responsibility models, in: Proceedings of the 21st International Conference on Advanced Information Systems Engineering, 2009, pp. 515–529, (Amsterdam).
[25] Munindar P. Singh, An ontology for commitments in multiagent systems: Toward a unification of normative concepts, Artificial Intelligence and Law 7 (1999) 97–113.
[26] Munindar P. Singh, Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects, in: Fabio Paglieri, Luca Tummolini, Rino Falcone, Maria Miceli (Eds.), The Goals of Cognition: Essays in Honor of Cristiano Castelfranchi, College Publications, London, 2012, pp. 1–29.
[27] Peter Checkland, Soft systems methodology: A thirty year retrospective, Systems Research and Behavioral Science 17 (2000) S11–S58.
[28] Joseph Goguen, Requirements engineering as the reconciliation of technical and social issues, in: M. Jirotka, J. Goguen (Eds.), Requirements Engineering: Social and Technical Issues, Academic Press, 1994, pp. 165–200.
[29] Yudistira Asnar, Paolo Giorgini, Fabio Massacci, Nicola Zannone, From trust to dependability through risk analysis, in: Proceedings of the Second International Conference on Availability, Reliability and Security, 2007, pp. 19–26.
[30] Paolo Giorgini, Fabio Massacci, John Mylopoulos, Nicola Zannone, Requirements engineering for trust management: Model, methodology, and reasoning, International Journal of Information Security 5 (2006) 257–274.
[31] Sara Jones, Marc Wilikens, Philip Morris, Marcelo Masera, Trust requirements in e-business, Communications of the ACM 43 (12) (2000) 81–87.
[32] Charles Haley, Robin Laney, Jonathan Moffett, Bashar Nuseibeh, Using trust assumptions with security requirements, Requirements Engineering 11 (2006) 138–151.

Elda Paja is a PhD student at University of Trento, Italy, under the supervision of Prof. Paolo Giorgini. She is currently involved in the FP7 EU-sponsored research project Aniketos, Ensuring Trustworthiness and Security in Service Composition. Her research interests include requirements engineering, conceptual modeling, socio-technical systems, and trust.

Amit K. Chopra is a Lecturer in Software Engineering at Lancaster University in the UK. His research interests lie broadly in the software engineering of sociotechnical systems with a special emphasis on modeling requirements and architecture.

Paolo Giorgini is associate professor and head of the Software Engineering, Formal Methods and Security group at the Department of Engineering and Computer Science of University of Trento. He received his Ph.D. degree from University of Ancona (Italy) and then he joined the University of Trento as assistant professor. He has worked on the development of requirements languages and the application of agent and goal-oriented techniques to (security) software analysis. He is co-editor in chief of the International Journal of Agent-Oriented Software Engineering and his publication list includes more than 180 refereed journal and conference proceedings papers and twelve edited books.