From Social Machines to Social Protocols: Software Engineering Foundations for Sociotechnical Systems

Amit K. Chopra
School of Computing and Communications
Lancaster University
Lancaster, LA1 4WA, UK
a.chopra1@lancaster.ac.uk

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
mpsingh@ncsu.edu

ABSTRACT

The overarching vision of *social machines* is to facilitate social processes by having computers provide administrative support. We conceive of a social machine as a sociotechnical system (STS): a software-supported system in which autonomous principals such as humans and organizations interact to exchange information and services. Existing approaches for social machines emphasize the technical aspects and inadequately support the meanings of social processes, leaving them informally realized in human interactions. We posit that a fundamental rethinking is needed to incorporate accountability, essential for addressing the openness of the Web and the autonomy of its principals.

We introduce Interaction-Oriented Software Engineering (IOSE) as a paradigm expressly suited to capturing the social basis of STSs. Motivated by promoting openness and autonomy, IOSE focuses not on implementation but on *social protocols*, specifying how social relationships, characterizing the accountability of the concerned parties, progress as they interact. Motivated by providing computational support, IOSE adopts the accountability representation to capture the meaning of a social machine's states and transitions.

We demonstrate IOSE via examples drawn from healthcare. We reinterpret the classical software engineering (SE) principles for the STS setting and show how IOSE is better suited than traditional software engineering for supporting social processes. The contribution of this paper is a new paradigm for STSs, evaluated via conceptual analysis.

1. INTRODUCTION

We take as our point of departure the vision of a *social machine*, conceived of in the early days of the Web [2], and which has been gaining prominence. The underlying idea is that a social machine represents a collective of humans and computers (or algorithms) working collaboratively on some problem [3, 18, 31]. In a nutshell, a social machine comprises humans and computers; it seeks to flexibly support social processes that underlie an open society via administrative assistance from computers [2, pp. 172–175].

In the recent literature, the distinguishing features of a social machine are a large numbers of users, social interaction among them, and how the data generated by their interactions helps solve the problem at hand. A typical social machine comprises Twitter and its users and serves purposes such as earthquake notification and prediction [33], traffic routing [27], and journalism [21].

We define a *sociotechnical system (STS)* as a system of *principals*—social entities such as people—interacting with each other with support from technical components for computing and communication. A social machine is thus an STS. A crucial limitation of current approaches lies in the fact that they leave the social and technical components, in effect, disconnected from each other. Specifically, the technical component is an implementation of some generic "lowest-common denominator" functionality such as text messaging or photo sharing. And, the social component is epiphenomenal: left purely to what the users make of their interactions. Not surprisingly, such lack of structure interferes with interoperability and thus subverts the vision of social machines interacting with one another to support complex social processes. Just as metadata and semantics are invaluable in linking data and services so too are they essential for supporting social processes.

To construct the big picture and formulate an effective solution, it helps to understand recent research on cybersecurity. Whereas traditional research on cybersecurity has been concerned with securing technical resources, such as computing and storage devices and communication networks, it has become clear that the most insidious challenges of cybersecurity arise at the human and social levels. To this end, it is important to recognize that any use of computing can be understood as a sociotechnical system. The social aspects of cybersecurity are integral concerns for social machines. For example, we would want to ensure that a social machine does not (depending upon its purpose) produce disinformation, malicious advice, or improper resource allocations.

In effect, what unites the concerns of social machines and the social aspects of cybersecurity is accommodating openness and autonomy. The problem in each case is of governance of an STS [37]. The central idea that we develop is that of, first, placing interaction front and center and, second, of modeling interaction in terms not of the technical but of the social elements of an STS. This idea brings us to the conception of accountability as a way of characterizing "good behavior" for each of the principals involved. Account-

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
DOI: http://dx.doi.org/10.1145/2872427.2883018.

ability here is directed from one principal to another and reflects the legitimate expectations the second principal has of the first. Thus, accountability is a relationship between two principals who are notional peers and characterizes the social machine we are considering.

In this spirit, we introduce an alternative paradigm called IOSE (pronounced ee-oh-zay, as in Italian) for *Interaction-Oriented Software Engineering*. IOSE is expressly suited to capturing the social basis of STSs. To promote openness and autonomy, IOSE focuses not on implementation but on *social protocols*, which specify how social relationships among the concerned principals progress through their interactions. To sustain improved computational support, IOSE posits that the meanings of social interactions be represented formally. These meanings on the one hand characterize the social relationships of the principals and on the other hand the state of the social machine being specified. Although IOSE as a paradigm is not limited to particular social relationships, for concreteness, our examples consider a few well-established types of social relationships, such as committing to perform an action, dialectically committing to the truth of some assertion, prohibiting another from performing an action such as divulging private information, and establishing an interpersonal relationship such as friendship.

As we conceive of it, first, to specify a social machine is to specify its intended social protocol. A protocol in our vision specifies all and only the relevant *social expectations* and the concomitant *accountability* of the principals participating in a social machine. That is, any expectation that makes a difference in a social machine must arise in the protocol. No operational detail must feature in the protocol unless it is part of some expectation. Second, to participate in the social machine is to enact that social protocol. In general, each social entity that participates in a social machine may apply its own software implementation to assist in its participation. The implementation would capture the policies by which its principal acts in a social machine. We can judge the correctness of such a software implementation with respect to the protocol in which it would participate.

An important consequence of adopting a protocol as a specification is that whereas a protocol provides a standard of correctness, it does not enforce compliant behavior. As an autonomous party, a principal may violate a protocol. In general, well-designed social machines (protocols) would be resilient against certain kinds of violations. For example, a protocol may incorporate means such as reputation, social censure, or economic penalties to sanction violators.

Contributions. We make the following contributions.

- *Gap analysis.* We analyze the foundational architectural model at the heart of traditional SE to demonstrate its inadequacy for supporting secure collaboration through social machines. In a nutshell, traditional SE neither accommodates autonomy nor provides a computational basis for accountability.
- *IOSE as the solution.* We show via conceptual analysis (using examples that typify a wide variety of settings) how IOSE accommodates autonomy and accountability via social expectations and thereby is equipped to support the engineering of social machines. We reinterpret in IOSE the classical SE principles of modularity, encapsulation, abstraction, and separation of concerns, yielding guidelines for any methodology for social machines. We evaluate prominent methodologies vis à vis these principles, showing where they fall short, thereby establishing the novelty of IOSE.
- *Benefits.* We show that an explicit treatment of social protocols and accountability promotes properties such as openness and innovation that motivate social machines.

We demonstrate the concepts, principles, and benefits of IOSE via a public health scenario.

Organization. This paper is organized as follows. Section 2 explains the limitations of traditional SE. Section 3 lays out the IOSE concepts and how systems are engineered following IOSE. Section 4 introduces the fundamental principles of IOSE. Section 5 evaluates existing approaches vis à vis IOSE. Section 6 relates IOSE to extant and envisioned social machines. Section 7 summarizes our contributions and Section 8 lays out an agenda of future research.

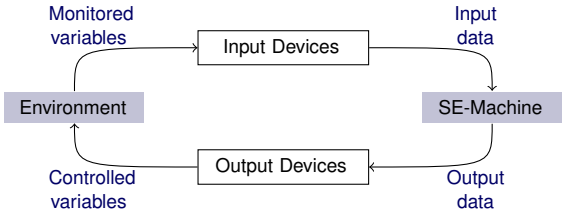
2. CURRENT SOFTWARE ENGINEERING

To avoid terminological confusion, we use the term *SE-machine* to refer to a software machine in the sense currently used in SE. The limitations of SE arise from the fact that it seeks to specify and deploy an SE-machine [48] that when implemented and installed, would satisfy stakeholder requirements. As we explain below, an SE-machine unreasonably limits principal autonomy and imposes an arbitrary standard of correctness that is disconnected from the expectations of the principals.

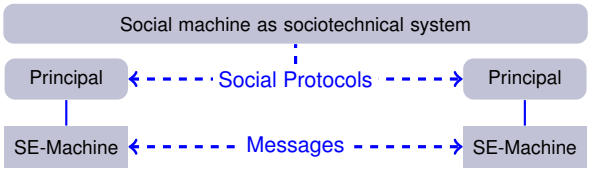
2.1 SE-Machines

SE seeks to produce a specification of software conceptualized as an SE-machine that would be a component in the system. For illustration, let us sketch van Lamsweerde's [43] influential account of SE. The *system as is* is the system with limitations identified. SE seeks to engineer the *system to be*, whose *objectives* are to avoid those limitations. The idea is to come up with a set of *services*, *constraints*, and *assumptions* under which the stated objectives would be met—some by the *software to be* as part of the *system to be*; some via responsibilities assigned to components in the *environment*, namely, people, devices, and existing software. The software to be is an SE-machine that maps inputs from its environment to outputs or effects in the environment by monitoring and setting appropriate variables. Figure 1a [43] depicts the SE-machine-environment configuration. To users in the environment, an SE-machine provides computational services via a user interface or API. The SE-machine mediates the interactions between its users and is thus a conceptually central entity even when it has a distributed implementation.

More generally, given a set of requirements, SE seeks to produce an *SE-machine* (i.e., *software to be*) specification that along with domain assumptions satisfies the requirements [48]. Existing requirements-based approaches, though differing in their details, instantiate the same concepts: SE-machine, environment, *system as is*, and *system to be*. In Tropos [5] and i* [46], the environment is a set of *actors* and the SE-machine a lone *system actor*. For example, Tropos would create a system actor for an entire healthcare system that would capture a consistent subset of the goals of all stakeholders. KAOS [15] would create a set of software components with designer-assigned individual goals. The components represent a possibly distributed SE-machine [48,



(a) Current software engineering is machine-oriented [43]: Given STS requirements, design an SE-machine to control the environment. The SE-machine is conceptually monolithic, even though it may be implemented as a distributed system.



(b) IOSE: Given STS requirements, design a social protocol. Principals enact the protocol via their respective SE-machines, which communicate by messaging. A principal's SE-machine interfaces with the environment, exactly as depicted in Figure 1a, and helps a principal participate in a social protocol.

Figure 1: IOSE introduces social protocols as the way for specifying decentralized social machines. In doing so, it goes beyond the machine-oriented conception of traditional SE.

p. 23]. Even though the components may be distributed, the idea that a single entity's perspective dominates is at odds with autonomy in our present setting of social machines.

2.2 Limitations of Machine Orientation

Let us apply the above conceptual model of systems to a healthcare scenario, specifically, immunization. Immunization stakeholders include government agencies, insurance companies, parents, physicians, healthcare practices, schools, universities, and the subjects themselves. The stakeholders would have requirements concerning universal coverage in the target population, schedules, safety of subjects and the general populace, exemptions, handling and administration of vaccines, record-keeping, costs, and so on.

Because of their relevance to public health, immunization requests must be dealt with promptly. Consider a single requirement, SchedWeek: physicians must schedule a parent's request for an immunization consultation within a week if a slot is available. (For brevity, we disregard auxiliary requirements such as that a physician be able to configure a daily schedule.) Suppose an SE-machine, M_{FCFS}, meets SchedWeek by implementing the first-come first-served (FCFS) policy. In doing so, M_{FCFS} displays implementation bias [48] and limits its user's (the principal's) autonomy: a physician who wants to meet SchedWeek but via another policy cannot do so, except by bypassing the SE-machine.

Implementation bias arises even if the SE-machine were the most flexible correct solution for a requirement. M_{FCFS} is such a solution for the requirement SchedWeekFCFS: physicians must meet SchedWeek and on an FCFS basis. Here, the SE-machine represents a *procedural idealization* of work [40]: principals must respect the SE-machine, which regiments their interaction, limiting autonomy and obscuring accountability. For example, a physician may want to break the formal SchedWeekFCFS requirement to contribute to emergency relief efforts after a natural disaster, which conflicts with both SchedWeekFCFS and SchedWeek. Traditional SE forces a formal ideal based on SE-machines, which may be overridden by users, possibly based on some informal ideal. Thus the machine may be subverted and there is no computational support for accountability.

The above analysis for a single requirement can be generalized to the entire set of immunization-related requirements. Following traditional SE, one would end up with an SE-machine that serves as the immunization software "platform" that provides a set of services, which would be accessible to users, possibly over the Web, depending upon the role they play in the social machine (e.g., school or parent). Such a platform would restrict the autonomy of its users and would at best represent a procedural idealization of work.

It is not that traditional SE does not concern itself with sociotechnical notions—approaches such as KAOS and Tropos clearly do. Our claim is more specific: current SE specifies, not the STS, but SE-machines that would reside in an STS. This formulation interferes with autonomy and openness and thereby restricts the innovation that motivates social machines. By contrast, IOSE specifies the STS itself in terms of the relevant social expectations.

3. IOSE CONCEPTS

In contrast to traditional SE, IOSE captures an ideal based on social expectations, which it gives computational status and whose progress principals can potentially track as they interact. Principals are accountable for such expectations, though free to violate them. Violating expectations is crucial for innovation. We elaborate below.

3.1 Social Protocols

In IOSE, the social expectations that hold in a social machine represent its *social state*. A *(social) protocol* for an STS specifies how its social state progresses as its principals interact. Traditionally, the protocols are informal and often take the form of a business contract or regulations that principals would have to comply with upon adopting different roles to conduct a social engagement. For example, North Carolina has regulations covering immunization that subjects, schools, universities, physicians, parents (or guardians for minors), and the health commission (as a representative of the State) must follow.

A social protocol as defined above serves three purposes. One, it makes explicit the social expectations of the principals in an engagement while giving them the flexibility to follow their individual goals. For example, schools are expected by the health commission to verify immunization, maintain records, and transfer records to other schools upon request. Two, the protocol identifies who is accountable to whom for what expectation. For instance, schools are accountable to the health commission for informing parents that immunization records for their children must be provided within 30 days of joining. Parents are accountable to the school for providing the records. Three, the protocol frees principals to implement their SE-machines, imposing only a standard of compliance. For example, a parent may

apply his or her own policy on whether to provide the child's immunization records within the 30-day window and, if so, whether to do so on a specific day in that window. The parent may in fact purposefully fail to comply. The protocol may specify penalties in case of noncompliance.

Accordingly, a key idea of IOSE is to specify a social machine via a protocol: one or more *roles*, their interactions, and the *social meanings* [8] of those interactions. The social meanings are specified in terms of social expectations. Figure 1b describes how IOSE applies. Principals adopt these roles to instantiate or participate in the desired social machine. The principals communicate with each other within the scope of the social machine, subject to the meanings defined in the protocol. As the principals interact, the state of the social machine progresses according to the protocol.

3.2 Compliance and Innovation

Figure 2 illustrates the broad IOSE method and artifacts. Stakeholders specify a protocol that meets their requirements. The protocol imposes requirements on any principal who would adopt any of its roles. Being autonomous, principals have their own requirements, and may respect or disregard protocol-imposed requirements. A principal's requirements would be implemented in its SE-machine.

Capturing social machine requirements as protocols is crucial in avoiding implementation bias. Although a principal may choose to acquire an off-the-shelf implementation of its chosen roles, the principals are decoupled with respect to their decisions on their implementations. Notice that the violation of a protocol requirement is not necessarily a bad thing. Sometimes proper performance requires stepping outside the bounds of the given contract or regulation. Such good violations can be thought of as sources of innovation: if effective principals repeatedly violate a social machine's protocol, then that is motivation to reformulate the protocol to reduce such violations. For example, a physician may exempt a child from immunization based on the presence of a contraindication that is not on the list of official contraindications and thus save a child's life. Later, that contraindication may be added to the official list.

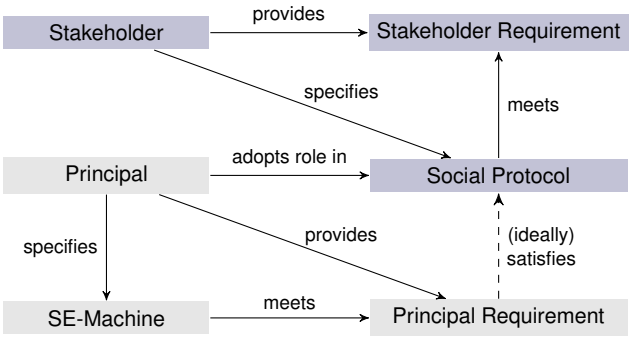


Figure 2: IOSE concepts underlying a social machine.

3.3 Normative Social Expectations

We introduce social protocols that specify social relationships via normative expectations such as commitments, authorizations, and prohibitions [36].

To this end, we extend the commitment protocols approach, which focuses on *practical commitments* [45]. A practical commitment captures an elementary social expectation between a debtor and a creditor [35]. Specifically, the commitment $\mathsf{C}(x, y, r, u)$ says that the debtor x commits to the creditor y that if the antecedent r comes to hold, then x will bring about the consequent u. The commitment represents the creditor's expectation of the debtor (that if r, then u); the debtor is accountable to the creditor for it. $\mathsf{C}(x, y, \top, u)$ represents an unconditional commitment. The social nature of commitments owes to the fact that they progress due to, not the principals' internal reasoning, but their interactions. Commitment protocols specify the meanings of *messages* in terms of how they affect commitments.

In contrast to a practical commitment, a *dialectical commitment* $\mathsf{D}(x, y, r, u)$ represents a claim made by x to y that if r holds, then u holds; it represents y's expectation of x that x's claim is true [42]. An *authorization* $\mathsf{A}(x, y, r, u)$ says that x is authorized by y to bring about u if r holds; it represents x's expectation of y that if r holds, then x will be able to perform u. A *prohibition* $\mathsf{P}(x, y, r, u)$ says that x is prohibited by y to bring about u if r holds; it represents y's expectation of x that if r holds, then u will not hold.

Table 1 shows a snippet of a social protocol for immunizations. It shows three messages with their meanings. Registering as a school (s) with the commission (c) means that the school practically commits to the commission that if a parent (p) registers a child (ch) with the school, then the school will request immunizations-related information for the child. Further, it means that the commission is authorized to audit the school. A parent registering a child with a school means that the parent commits practically to producing either the immunization records upon request by the school or an exemption provided by a physician (ph). An exemption amounts to a dialectical commitment of the physician to the parent that the child has contraindications that make immunization unsafe for the child. Finally, when a physician registers with the commission, he or she is prohibited by the commission from giving immunization information to agencies (ax) not on a list of (authorized) parties.

Table 1: A partial social protocol for immunization.

Message	**Meaning**
$\mathsf{regSch}(s, c)$	$\mathsf{C}(s, c, \mathsf{regCh}(p, s, ch), \mathsf{reqImm}(s, p, ch))$ $\wedge\ \mathsf{A}(c, s, \top, \mathsf{audit}(c, s, ch))$
$\mathsf{regCh}(p, s, ch)$	$\mathsf{C}(p, s, \mathsf{reqImm}(s, p, ch), \mathsf{prodImm}(p, s, ch)$ $\vee\ \mathsf{exemCh}(ph, p, ch))$
$\mathsf{exemCh}(ph, p, ch)$	$\mathsf{D}(ph, p, \top, \mathsf{hasContra}(ch))$
$\mathsf{regPh}(ph, c)$	$\mathsf{P}(c, ph, \mathsf{reqInfo}(ax, ph) \wedge \neg\mathsf{listed}(ax),$ $\mathsf{giveImmInfo}(ph, ax))$

Now we can consider possible enactments of the protocol in Table 1 in which principals adopt protocol roles and interact with each other. Table 2 introduces a situation where Alessia plays parent, Unity plays school, Health Commission (HCom) plays commission, and Bianca plays physician.

Figure 3 depicts a potential enactment (adopting the expectations defined in Table 2) and shows how the social state progresses with communication between the principals.

Specifically, the subject of immunization, that is, the child, is Paul. LI (short for Lancaster Insurance) is prohibited from receiving immunization information. Due to prior interactions, the initial state (top left) includes Unity's commit-

Table 2: Expectation instances in an enactment of Table 1's immunization social protocol. Here p (parent), s (school), c (commission), ph (physician), ch (child), and ax (party barred from immunization information) are bound to Alessia, Unity, HCom, Bianca, Paul, and LI, respectively.

ID	Commitment
e_0	C(Unity, HCom, regCh(Alessia, Unity, Paul), reqImm(Unity, Alessia, Paul))
e_1	A(HCom, Unity, $\top$, audit(HCom, Unity, Paul))
e_2	P(HCom, Bianca, reqInfo(LI, Bianca) $\wedge \neg$listed(LI), giveImmInfo(Bianca, LI))
e_3	C(Unity, HCom, $\top$, reqImm(Unity, Alessia, Paul))
e_4	C(Alessia, Unity, reqImm(Unity, Alessia, Paul), prodImm(Alessia, Unity, Paul)$\vee$ exemCh(Bianca, Alessia, Paul))
e_5	C(Alessia, Unity, $\top$, prodImm(Alessia, Unity, Paul) $\vee$ exemCh(Bianca, Alessia, Paul))

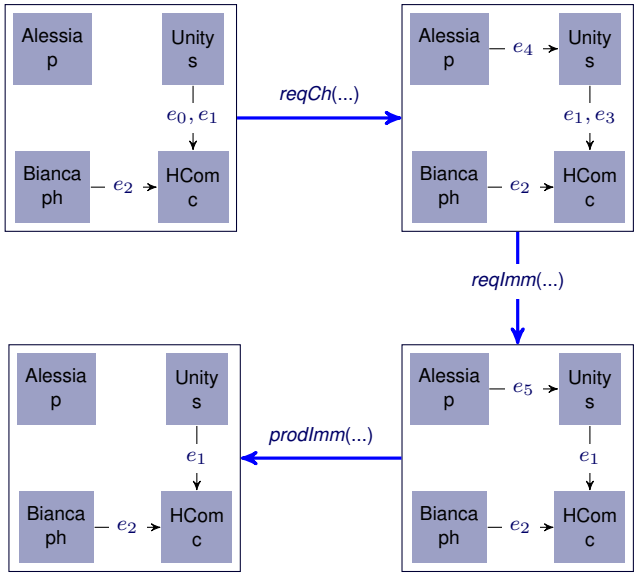


Figure 3: Social state progression in an enactment.

ment to HCom to require immunization of children registered there (e_0), HCom's authorization to conduct an audit (e_1), and the prohibition on Bianca on releasing information to LI (e_2). Alessia's registering Paul at Unity leads to the state (top right) where Alessia is committed to Unity to presenting Paul's immunization information upon request (e_4) and Unity is unconditionally committed to HCom to requesting immunization information from Alessia (that is, e_0 is detached and in its place we have e_3). Unity requests immunization information from Alessia, leading to the state (bottom right) where e_3 is discharged (not shown) and e_4 is detached resulting in Alessia's unconditional commitment to provide either Paul's immunization information or an exemption certificate (e_5). Alessia provides the information, leading to the state (bottom left) where e_5 is discharged. Only e_1 and e_2 remain in this state.

In IOSE, the state of a social machine is explicit and can be computed from the protocol if all messages are observed. The social state does not rely upon the principals' beliefs or goals, or their SE-machine implementations. For example, even if Alessia implements her SE-machine to ignore immunization information requests from the school, she would be committed by fact of her participation in the protocol as a parent and the messages she previously exchanged.

3.4 Principals' SE-machines

Now we turn to how a principal can implement its SE-machine in order to participate in a social protocol. Let us discuss some possible implementations of Unity's and Bianca's SE-machines for the social protocol of Table 1. The protocol imposes two requirements on Unity's SE-machine: that (1) it ask for immunization information upon registration; and (2) grant HCom access to immunization information for audit purposes. Unity has additional requirements besides those imposed by the protocol. For example, Unity would like to notify its internal medical office and the child's tutor if a child does not have the requisite immunizations. In addition, in case of an exemption from a physician that notes a nonstandard contraindication, the school would like to consult with the physician. Listing 1 shows an SE-machine listing (in pseudocode) that meets these requirements.

Listing 1: A simplistic SE-machine for Unity.

```
public void uponEvent(e){

 if (e == reqSch(commission))
   authorizeImmAccess(database, commission))

 if (e == regCh(parent, child))
   if (immInfo(database, child) is incomplete)
     send(reqImm(parent, child))

 if (e == prodImm(parent, child, info))
   if (info is complete)
     clearForClasses(child)
   else
     suspendEnrollment(child)
     notifyMedicalOffice(child, "INCOMPLETE")
     notifyTutor(child, "INCOMPLETE")

 if (e == exemCh(parent, child, physician))
   if (contraindication is valid)
     clearForClasses(child)
     notifyMedicalOffice(child, contraindication)
   else
     consult(physician, child)
}
```

Listing 2 shows an SE-machine for Unity that is potentially noncompliant with the immunization protocol because it may lead to the commitment to request records being violated. Instead of requesting immunization records from the parent as is required by the protocol, this SE-machine checks if the child is more than 14 years of age and if the parent has confirmed that the child has had all the immunizations.

Listing 2: An alternative SE-machine for Unity.

```
public void uponEvent(e){
  ...
  if (e == regCh(parent, child))
    if (immInfo(database, child) is incomplete)
      if (age(child)>14 and confirmsImmunization(parent))
        clearForClasses(child)
  ...
}
```

Listing 3 shows a potential implementation for Alessia's SE-machine. In general, each principal's SE-machine would help that principal participate in the social machine.

Listing 3: An SE-machine for Alessia.

```
public void handleEventOccurences(e){
  if (e == reqImm(school, child)))
     info = searchPersonalDB(child, "immunization")
     if (info not found)
       send(Apology(school, child, "Sorry, no information''))
     else
       send(prodImm(school, child))
     remindToCall(school, 2 days, "note:immunization")
}
```

3.5 Benefits of IOSE for Social Machines

The above exercise demonstrates how, in realizing social machines, IOSE respects principal autonomy and provides a standard for compliance. What would previously have been seen as the problem of specifying an immunization SE-machine turns, under IOSE, into two independent problems of (1) specifying an immunization social protocol, and (2) specifying principals' SE-machines, e.g., as demonstrated above for Unity and Alessia. This split reflects Parnas' conception of modularity as the division of labor [28].

The aforementioned split is key to principals being able to exercise their autonomy as they participate in an STS engagement, as we illustrate by specifying alternative SE-machines for Unity (Listings 1 and 2). The protocol, however, specifies what Unity as a school would be accountable for—regardless of the SE-machine it deploys. Further, in IOSE, the social expectations being explicit can potentially be reasoned about at runtime, enabling us to conceive of a progressing social state, as Figure 3 illustrates.

4. PRINCIPLES OF IOSE

We now introduce the core principles of IOSE, relating them to the classical principles of SE—modularity, abstraction, separation of concerns, and encapsulation. Although some intuitions behind the classical principles hold, IOSE yields principles that contravene those of traditional SE.

4.1 Accountability Modularity: Embedding in the Social World

In traditional SE, modularity refers to the decomposition, typically by functionality, of a system into components. A benefit of modularity is improving composability and taming complexity. The decomposition of a system (Figure 1a) into an environment and an SE-machine is a form of modularity. However, neither the environment nor the SE-machine can be accountable to anyone since they are not social entities.

IOSE characterizes a social machine via its roles and their mutual social expectations and accountability. In IOSE, it makes little sense to ask what functionality a role provides (a question one may sensibly ask for SE-machines, such as Web services); it makes sense though to ask though to whom and for what is a role accountable (a question that makes little sense for Web services). A social protocol essentially describes how a principal playing a role would be embedded in the social world by way of accountability.

Example. Alessia's communication (regCh) creates a commitment to Unity to provide immunization records upon request. This makes Alessia accountable to Unity for producing the records upon request. It makes no sense to say Alessia provides the functionality of producing immunization records because Alessia may decide not to do so.

Benefit. Promotes autonomy by not unduly restricting a principal's courses of action. Promotes accountability by providing a basis within a protocol for ensuring *correctness*: a principal acquires expectations by adopting a role in a protocol. A principal who fails to comply with its expectations would be accountable to the counterparties of the expectations. Sanctions would often be specified as part of the protocol. For example, the protocol may require the debtor of a violated commitment to compensate the creditor [11]. Alternatively, a sanction might be to eject a principal who repeatedly violates expectations—analogous to closing a member's account on eBay. Or, a sanction might involve escalation to a higher authority, as in litigation. Recall from Figure 1b that a protocol specifies a social machine and thus should incorporate any applicable sanctions.

A principal may apply a sanction that is not specified in the protocol. For example, Unity may avoid business with Alessia or Alessia may bad mouth Bianca to others. Such sanctions might themselves violate other social expectations, for which the sanctioning principal may be accountable. For example, Unity may be fined for treating Alessia unfairly.

4.2 Abstraction: Social Meaning as Central

Abstraction refers to the level of the concepts used in a specification. The ideal abstraction is sufficiently high-level to hide details and reduce complexity, yet sufficiently low-level to support drawing the necessary conclusions. A low-level abstraction would be a state machine or a sequence diagram, which are operational and lack social meaning. Abstractions such as goals and goal dependencies [5] are higher level but they too lack social meaning, as we explain in Section 4.4. In contrast, IOSE emphasizes abstractions that capture the meanings of interactions in a social machine.

IOSE requires making all social expectations explicit in the protocol, including stating the operational meaning of each communication in terms of how it affects the state of a social machine, as Figure 3 illustrates.

Example. Table 1 specifies the message meanings in an immunization protocol.

Benefit. Promotes accountability and loose coupling. If the meanings are explicit, the principals can potentially check their own and each other's compliance with respect to a social machine. Further, absent explicit meanings, interoperability becomes difficult. For example, schools and parents may come to different conclusions about what their mutual expectations and accountabilities are. Moreover, an explicit meaning helps keep the implementations of the various principals separate and thus avoids hidden couplings between their implementations.

IOSE posits that any requirement characterizing a social machine must take the form of an expectation in the social protocol. That is, some role must be accountable for each requirement. IOSE forbids global constraints. Expressing any global constraint causes two problems. First, it is not clear who is accountable for it. Second, it presumes that the constraint can be enforced without any principal deciding to enforce it, since no one is accountable for it, which involves infringing upon the autonomy of the concerned principals.

For example, specifying that prodImm must follow reqImm leaves no one accountable for ensuring it. Is Alessia at fault for not delaying sending prodImm or is Unity at fault for not sending reqImm early enough? Instead, if (say) the infrastructure were to enforce the constraint, it could violate either Alessia's or Unity's autonomy. Even an explicit environmental assumption about the infrastructure enforcing the constraint would not capture who is accountable.

In contrast, IOSE would tackle it through the social protocol involved. We would introduce an INFRASTRUCTURE PROVIDER role who would be accountable for this requirement. It could achieve it by controlling the infrastructure suitably. The other roles would interact with INFRASTRUCTURE PROVIDER and lack privileges that conflict with the infrastructure's configuration. In general, an agent playing a role that is responsible for a requirement may achieve it by persuading other agents to act accordingly.

Example. The above ordering constraint can be expressed as $\mathsf{C}(p, s, \top, \mathsf{reqImm}(\ldots)$ precedes $\mathsf{prodImm}(\ldots))$, assuming a suitable formalization of "precedes" [23].

Benefit. Promotes autonomy and accountability. Expressing a constraint to be enforced by an SE-machine hides accountability. For each constraint, some principal ought to be accountable for it, possibly as the operator of the concerned SE-machine.

4.3 Separating Social and Technical Concerns

Separation of concerns refers to the treatment of each aspect of a problem independently of, *yet* in relation to, others. In software engineering practice, this principle echoes separation of policy and mechanism [20].

For social machines, we must distinguish principals from technical entities (e.g., resources, software components, and infrastructure) that they own, control, or access. Social expectations are meaningful only among principals, who alone are autonomous and accountable: a patient cannot sue a needle but can sue a nurse or a needle manufacturer.

Example. Figure 3 shows Unity's and Alessia's social expectations of each other. Each of them controls an SE-machine, as in Listings 1 and 3, respectively. However, these SE-machines are not socially visible.

Benefit. Clarifies relationships among the entities; ascribes accountability only to principals, who may develop and operate technical entities.

4.4 Encapsulation: No Principal Internals

Encapsulation refers to the principle that a module reveal no more information than is necessary to effectively use it, in particular, that it reveal no implementation details.

In IOSE terms, encapsulation maps to the idea that a social machine cannot be specified in terms of mental abstractions such as beliefs or goals of its stakeholders or principals. The mental abstractions are not observable. In particular, each protocol role refers only to the social expectations resulting from the communications that a principal adopting it would be involved in, not anyone's mental state.

Example. None of the roles of the immunization social protocol have any individual or joint goals of any sort, not even of getting anyone immunized. However, Unity and Alessia may have goals and their SE-machine may encode them.

Benefit. Promotes loose coupling by hiding details not relevant to the interaction. Promotes accountability by ensuring the social state is based exclusively on the protocol and the same observations. This contrasts with the mental concepts—a protocol cannot dictate what goals and beliefs a party may adopt. The social perspective is essential for interoperability and compliance checking [34].

4.5 STS Configurations

Figure 4 contrasts three architectural configurations to help explain how IOSE differs from traditional SE. In each picture, X and Y are autonomous principals and Communication refers to the infrastructure via which they interact.

Figure 4a shows the STS setting that predates IT or uses IT only for transport, e.g., if communication is via foot messenger or email. The principals are autonomous, may follow some protocol, and have an understanding of how their expectations progress. For example, a parent may request a physician for an immunization record and the physician may send back the record along with an invoice. The message format may be formalized but the social meaning of the protocols is not represented formally and unwritten conventions or natural language descriptions govern the interactions.

Figure 4b shows how an SE-machine, designed following traditional SE, sits between any two principals and helps realize their STS engagement. The SE-machine is a technical entity but interferes with the social interactions of X and Y by forcing them to comply with whatever standard of interaction is implemented in the SE-machine.

Figure 4c illustrates IOSE: unlike in Figure 4a, the protocol explicitly specifies the social interactions and omits the technical details. Specifically, the protocol describes how each message affects expectations among the principals playing different roles. In a possible implementation, the interactions of the principals may be supported via an *expectations middleware* that can track on behalf of a principal the expectations concerning that principal in the social machine. The middleware is a domain-agnostic implementation of expectations reasoning, analogous to the implementation of HTTP in clients and servers being agnostic as to applications. In IOSE, protocols are the essential domain-specific construct. By contrast, in 4b, the SE-machine is the essential domain-specific construct.

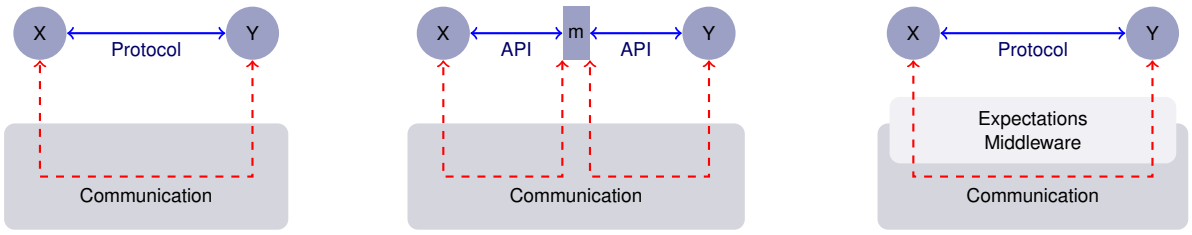
Note that Figures 4a and 4c support configurations where principals interact via a "central" principal, as in engagements such as business brokerages or escrow. Such engagements may be necessitated by stakeholder requirements. For example, eBay (the company) mediates interactions between buyers and sellers on eBay's website. However, such configurations differ from Figure 4b, where it is a technical and not a social entity that mediates interactions. Recall the requirement SchedWeekFCFS from Section 2. Capturing it in the protocol as a commitment from physicians to parents that physicians decide how to handle yields Figure 4c. Instead, building an SE-machine for it yields Figure 4b.

5. PROMINENT SE APPROACHES

We describe and evaluate some prominent SE approaches, especially their underlying models, with respect to the IOSE principles. We choose approaches that are representative of major classes of modeling approaches emphasizing requirements, agents, and services, respectively.

5.1 Tropos and i*

Tropos and i* emphasize requirements modeling and analysis. They model stakeholders as actors and requirements as



(a) Transport. (b) SE-machine oriented. (c) Interaction oriented.

Figure 4: A historical perspective on protocols and interaction. (Circles are principals; rounded rectangles are infrastructure; the rectangle is a resource. Dashed lines indicate message paths; solid lines indicate connectivity at the level of the specification.) In *interaction oriented*, each principal is supported by its own SE-machine, as in Figure 1b.

goals and support relational requirements as dependencies between actors. For example, x depends on y for its goal p if y is *able* to achieve p and *intends* to deliver p.

- *Accountability Modularity:* Violated. It is not meaningful to talk of the system actor's, an SE-machine's, accountability as there is no principal behind it.
- *Explicit Social Meaning:* Violated. Dependencies are meant to provide high-level abstractions to capture interactive relations, but refer to actors' mental states, and lack social meaning.
- *No Global Constraints:* Violated. Extensions of i* introduce global temporal constraints, e.g., that one goal should be achieved before another [22].
- *Separating Social from Technical:* Partially fulfilled. Although i* treats social and technical actors alike, in early requirements modeling, all actors are stakeholders; however, a system actor is introduced beginning from late requirements modeling.
- *No Principal Internals:* Violated. Dependencies between actors are rooted in the internals of the actors, such as goals, intentions, and procedures.

KAOS, which is also goal-oriented, violates the IOSE principles; for brevity, we omit a detailed discussion of it.

5.2 Gaia

Zambonelli et al.'s Gaia [47] is an early agent-oriented methodology that takes an organizational approach.

- *Accountability Modularity:* Partially. Although Gaia has notions such as responsibility (capturing what an agent *ought* to do), that is not the same as accountability (what others expect an agent to do).
- *Explicit Social Meaning:* Partially fulfilled. Gaia supports specifying interactions among roles, but not the meanings of communications.
- *No Global Constraints:* Violated. In Gaia, roles are specified via control flow abstractions, e.g., that one activity should follow another.
- *Separating Social from Technical:* Partially fulfilled. Gaia supports social constructs by modeling systems as organizations specified via roles. However, an agent in Gaia is any object with a thread of control, e.g., a mail client—that is, an SE-machine.
- *No Principal Internals:* Violated. Gaia places internal activities in role specifications, thereby exposing an agent's internal decision-making.

5.3 Choreography

The earliest coordination models for Web services were based on workflows [4]. OWL-S [26] supported specifying workflows enriched with Semantic Web-style service annotations to aid discovery and composition. However, the notion of a workflow is a centralized one: it represents a single locus of control even when the services involved are distributed [39]. The notion of choreography represents an alternative to workflow-based coordination. A choreography specifies messages among roles and constraints on message ordering and occurrence [29]. Choreographies may be specified using control-flow constructs, e.g. in WS-CDL [44] or in a more declarative language such as temporal logic [25]. The benefit of specifying a choreography is that it naturally supports multiple loci of control. Naturally, choreographies appear to fit the paradigm of Figure 4a. The main shortcoming of choreographies is that they focus on abstractions for messages ordering and occurrence, not on high-level social expectations as IOSE does. Somewhat surprisingly, choreographies fail the principle of encapsulation.

- *Accountability Modularity:* Violated. Choreographies do not specify the accountabilities of the roles involved.
- *Explicit Social Meaning:* Violated. Lack a representation of social meaning.
- *No Global Constraints:* Violated. Global constraints on message ordering and occurrence are central to a choreography.
- *Separating Social from Technical:* Fulfilled. Choreographies specify interactions with reference to roles that principals may adopt.
- *No Principal Internals:* Partially violated. Ideally, choreographies specify constraints on messaging and nothing else (for example, Dijkman and Dumas [16] are quite explicit about this). However, in practice, choreographies specify the internals of principals as well. For example, choreographies in WS-CDL may specify variables local to a role and the role's decisions based on that variable, e.g., when to end negotiation of the price of some goods [44]. In the formalization of a tax filing scenario, Mendling and

Hafner [24, p. 532] specify a tax adviser's internal compliance checks in handling a client's annual statement, thereby violating encapsulation.

Much of the high-level modeling work in service-oriented computing points towards an implicit machine-oriented mindset, especially because these works tend to violate the principle of encapsulation. For example, Gordijn et al. [17] produce Web services—conceptually, SE-machines—from high-level e^3-value models.

5.4 Summary of Evaluation of SE Approaches

Table 3 says that SE approaches poorly support accountability and therefore are inadequate as bases for social machines as understood in this paper.

Table 3: Evaluation: ✓, −, and × denote fulfilled, partially fulfilled, and violated, respectively.

Principle	Tropos	Gaia	Chor.	Protocols
Accountability modularity	×	−	×	✓
Explicit social meaning	×	−	×	✓
No global constraints	×	×	×	✓
Separating social	−	−	✓	✓
No principal internals	×	×	−	✓

6. SOCIAL COMPUTING

Let us now discuss the relevance of IOSE to social computing as characterized both by extant (mainly Web 2.0 applications) and envisioned social machines.

6.1 Extant Social Machines

Current social machines are Web 2.0 applications incorporating social relationships such as citing, following, friendship, having a crush on, managing, being physician of, being customer of, being mentor of, kinship, common affiliation, and affinity. A social relationship maps to one or more social expectations that help characterize a social machine. Some social expectations, such as commitments and prohibitions, may have wide applicability across many domains. Others, such as *friend* and *contact*, may be limited to specific domains. IOSE applies for any of them, the particulars of satisfaction, violation, and sanctioning being specific to each social machine.

IOSE helps understand conventional social machines based on Facebook and Wikipedia via social expectations. When a Facebook user Elisa adds Franco as a *friend*, Elisa authorizes Franco to view her status. When Pietro *likes* Luca's post on Facebook, he creates an expectation among his friends that he endorses it (a dialectical commitment). Wikipedia describes at length who may edit pages and how disputes must be recorded and resolved, including when disputed articles may be locked and users blocked. Expectations in Wikipedia may be modeled as commitments, prohibitions, and authorizations. IOSE supports abstractions for modeling expectations underlying potentially arbitrary social machines. For example, we may exploit the *friend* relationship to generate informal expectations for resource allocation, as Caton et al. [7] do.

Table 4 formalizes the interactions concerning status updates of a Skype-like messaging system. Here, x, y, z, and z' play role USER, and i plays INTERMEDIARY, such as the Skype organization. The relation $\mathsf{CON}(x, y)$ means that y is on x's contact list. *addContact*(x,i,y) is a message from x to i that means that y is on x's contact list and, further, x authorizes i to show x's status to y if x is on y's contact list; *acceptContact*(y,i,x) is a message from y to i—meaning that x is on y's contact list and y authorizes i to show y's status to x if x has authorized i to do so. If y accepts, i infers that i is now unconditionally authorized by x to show x's status to y. From this, i further infers that i is unconditionally authorized by y to show y's status to x. Table 4 in addition specifies the meaning of deleting a contact, which ensures that if either of the users involved deletes the contact, both users' authorizations to the intermediary are revoked.

Table 4: A partial status sharing social protocol

Message	Meaning
addContact(x,i,y)	$\mathsf{CON}(x, y) \wedge \mathsf{A}(i, x, \mathsf{CON}(y, x), \mathsf{showStatus}(x, y))$
acceptContact(y,i,x)	$\mathsf{CON}(y, x) \wedge \mathsf{A}(i, y, \mathsf{A}(i, x, \top, \mathsf{showStatus}(x, y)), \mathsf{showStatus}(y, x))$
deleteContact(z,i,z')	$\neg\mathsf{CON}(z, z') \wedge \neg\mathsf{A}(i, z, \top, \mathsf{showStatus}(z, z'))$

It is worth considering the architecture of today's social machines: Is it captured by Figure 4b or by Figure 4c? Current research treats social machines, e.g., those based on Twitter, as centered on a social media service that users employ toward whatever purpose they may have, thus pointing toward Figure 4b. Specifically, a Twitter social machine is not based on explicit social expectations and as such offers no computational support for them. Expectations and accountability are determined offline in ad hoc ways. It may be argued that a Twitter social machine encourages creativity because "anything goes." The problem, of course, is that "anything goes" is not a viable social constraint—including of the kind that Berners-Lee [2] alluded to—and does not support accountability. Notably, concerns about Twitter in settings such as disaster response [41] are rooted in concerns about accountability. IOSE addresses the challenge of fostering creativity by providing a standard of correctness based on expectations rather than implementation.

6.2 Social Machines as Envisioned

Let us discuss how IOSE relates to the key challenges in realizing social machines based on Web architectures as identified by Hendler and Berners-Lee [18].

Supporting creativity. Enabling people to act in creative ways is a major challenge. IOSE promotes creativity by specifying social constraints, as Hendler and Berners-Lee motivate, via expectations in social protocols. In IOSE, the constraints may be violated, thereby fostering creativity, as Sections 2 and 3 emphasize. For example, in the immunization scenario, innovation arises from physicians noting new contraindications; schools giving parents leeway in meeting their requirements based on contextual information; and public health agencies noting novel epidemiological patterns. Such innovations may violate the current social protocol but could be incorporated as enhancements in a revised protocol.

Reducing administrative burden. IOSE facilitates providing administrative support for social machines through

the expectations middleware representing changing social state, as illustrated in Figure 4c. The middleware would track expectations and compliance, thereby promoting interoperability and accountability. Key performance indicators based on the satisfaction of expectations would be crucial for effective administration.

Information accountability and trust. IOSE would address the challenge of information accountability by modeling the requisite expectations and reasoning about them during enactment. For example, the immunization scenario demonstrates commitments, prohibitions, and authorizations. Such primitives can be used to capture security and privacy requirements in a wide variety of settings. In addition, IOSE provides a social foundation for trust based on social expectations and accountability: doing so is crucial in providing a semantic basis for trust as opposed to much of current research, which concentrates on metrics but hides the meanings underlying links and interactions.

Semantics and interoperability. What abstractions and representations, especially of real-world entities, would promote interoperability to support complex social machines? IOSE provides a possible answer through abstractions that capture the meaning of social processes via social expectations. This explicit meaning facilitates interoperability by representing the essential social-level coupling (e.g., via social norms and other expectations) between autonomous principals while excluding any technical-level (i.e., based on SE-machines) coupling between them.

7. DISCUSSION

The social machines vision is centered on the idea of achieving secure collaboration, especially when we recognize security as integrally involving human and social aspects. Existing SE approaches are geared toward specifying an SE-machine. Because they deemphasize interaction, these approaches, including those that employ mental abstractions such as goals, fall short [8, 34] in capturing the essence of social machines. These approaches treat the SE-machine as the ideal, omitting a social notion of accountability.

In contrast, IOSE emphasizes autonomy and interaction and specifies a social machine as a social protocol. IOSE posits a computational notion of social state: the social expectations and concomitant accountabilities that hold among interacting principals. The protocol specifies how the social state progresses in a social machine. In this manner, IOSE addresses key challenges in realizing the vision of social machines: accommodating autonomy; promoting creativity; automating administrative tasks; and promoting accountability, trust, and interoperability.

A potential concern to IOSE is that social interactions are rich and therefore cannot be computationally represented. However, computing support must rely upon a computational representation. If the representation is about text messages, then the resulting computation can support only text messages, leaving all meaning as epiphenomenal. By providing high-level abstractions, IOSE lifts computations to the level of accountability and improves the effectiveness and predictability of collaboration. Additional, informal interactions can be placed on top of any protocol.

Modeling accountability does not alter the power structures in an organization or society: those depend primarily upon the governance in place. Arguably, an IOSE-based methodology would enhance transparency regarding who is accountable and thereby promote the value of openness on the web [6]. It is worth noting that accountability is not the same as blame, which is akin to a sanction and subsequent to deliberation over the relevant facts.

8. ONGOING AND FUTURE WORK

One, how can we efficiently compute with expressive social protocol languages? To facilitate effective decision making by a principal, how can we advise it regarding its accountabilities to others and other's accountabilities to it? What are the concomitant programming abstractions and tools? Our desired social middleware must ensure interoperability, which is nontrivial in a decentralized setting [13].

Two, what are suitable additional formal abstractions for social protocols, besides those we illustrate? Existing work on normative relationships, e.g., [38], provides a good basis but additional results are needed on representing and reasoning about a richer variety of social expectations. Algorithms for dealing with qualitative degrees or measures of satisfaction and violation is a major challenge.

Three, how can we specify social protocols that capture stakeholder requirements? Approaches such as Colaba [10] that capture stakeholder rationales via arguments appear promising as do approaches such as Rodríguez et al.'s [32] that mine reusable model patterns from crowds. Protos [9] relates protocols to an abstract requirements engineering process, although it does not represent stakeholder rationales. A combination of such techniques is what is needed.

Four, how can we take advantage of existing social computing and legacy infrastructure and applications? Leading themes include social sensing, collaborative filtering, data mining, and social network analysis [1, 19, 30]. How can we adapt existing techniques to derive aggregate metrics from social expectations, e.g., to compute trust? How can we represent and reason about expectations in information and event stores, e.g., building on emerging specification languages for commitments and other norms [12, 14].

Five, how can we support the *governance* of a social machine, incorporating (in addition to a social protocol) the social machine's stakeholder-driven evolution informed by how they participate and the outcomes they obtain? Understanding satisfaction and violation of expectations and their impact on a social machine is a prerequisite for supporting innovation in a reasoned manner. The first prerequisite is transparency so that principals can make informed decisions about the STS's governance. The second prerequisite is flexibility so that a social protocol incorporates ways, e.g., voting procedures, by which its participating principals could amend the protocol on the fly.

IOSE is about rethinking the principles of SE to support collaboration through accountability to promote openness and autonomy. Adopting IOSE leads to new foundational research problems in Web semantics, information processing, software engineering, and distributed computing—essential to realizing the social machines vision as social protocols.

Acknowledgments

Thanks to Fabiano Dalpiaz, Mike Huhns, Michael Jackson, John Mylopoulos, and Erik Simmons for discussions. Munindar Singh thanks the US Department of Defense for partial support under the Science of Security Lablet.

9. REFERENCES

[1] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer. Friendship prediction and homophily in social media. *ACM Transactions on the Web*, 6(2):9:1–9:33, 2012.

[2] T. Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. Harper Business, New York, 1999.

[3] A. Bernstein, M. Klein, and T. W. Malone. Programming the global brain. *Communications of the ACM*, 55(5):41–43, May 2012.

[4] BPEL. Business process execution language for web services, version 1.1, May 2003. www-106.ibm.com/developerworks/webservices/library/ws-bpel.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[6] B. Burgemeestre and J. Hulstijn. Designing for accountability and transparency: A value-based argumentation approach. In J. van den Hoven, P. E. Vermaas, and I. van de Poel, editors, *Handbook of Ethics, Values, and Technological Design: Sources, Theory, Values and Application Domains*, chapter 15, pages 1–28. Springer, Berlin, 2015.

[7] S. Caton, C. Haas, K. Chard, K. Bubendorfer, and O. Rana. A social compute cloud: Allocating and sharing infrastructure resources via social networks. *IEEE Transactions on Services Computing*, 7(3):359–372, 2014.

[8] A. K. Chopra, A. Artikis, J. Bentahar, M. Colombetti, F. Dignum, N. Fornara, A. J. I. Jones, M. P. Singh, and P. Yolum. Research directions in agent communication. *ACM Transactions on Intelligent Systems and Technologies*, 4(2):20:1–20:23, 2013.

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

[10] A. K. Chopra and M. P. Singh. Colaba: Collaborative design of cross-organizational business processes. In *Proceedings of the Workshop on Requirements Engineering for Systems, Services, and Systems of Systems*, pages 36–43. IEEE, 2011.

[11] A. K. Chopra and M. P. Singh. Specifying and applying commitment-based business patterns. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*, pages 475–482, 2011.

[12] A. K. Chopra and M. P. Singh. Cupid: Commitments in relational algebra. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, pages 2052–2059, Austin, Texas, Jan. 2015. AAAI Press.

[13] A. K. Chopra and M. P. Singh. Generalized commitment alignment. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multiagent Systems*, pages 453–461. IFAAMAS, 2015.

[14] A. K. Chopra and M. P. Singh. Custard: Computing norm states over information stores. In *Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 10 pages, Singapore, May 2016. IFAAMAS.

[15] A. Dardenne, A. V. Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.

[16] R. M. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach. *International Journal of Cooperative Information Systems*, 13(4):337–368, 2004.

[17] J. Gordijn, E. Yu, and B. van der Raadt. E-service design using i* and e^3value modeling. *IEEE Software*, 23(3):26–33, 2006.

[18] J. Hendler and T. Berners-Lee. From the semantic web to social machines: A research challenge for AI on the world wide web. *Artificial Intelligence*, 174(2):156–161, 2010.

[19] C. C. Kling, J. Kunegis, H. Hartmann, M. Strohmaier, and S. Staab. Voting behaviour and power in online democracy: A study of LiquidFeedback in Germany's Pirate Party. In *Proceedings of the Ninth International Conference on Web and Social Media*, pages 208–217, 2015.

[20] B. W. Lampson and H. E. Sturgis. Reflections on an operating system design. *Communications of the ACM (CACM)*, 19(5):251–265, May 1976.

[21] D. L. Lasorsa, S. C. Lewis, and A. E. Holton. Normalizing Twitter: Journalism practice in an emerging communication space. *Journalism Studies*, 13(1):19–36, 2012.

[22] S. Liaskos and J. Mylopoulos. On temporally annotating goal models. In *Proceedings of the 4th International i* Workshop*, pages 62–66. CEUR, 2010.

[23] E. Marengo, M. Baldoni, A. K. Chopra, C. Baroglio, V. Patti, and M. P. Singh. Commitments with regulations: Reasoning about safety and control in Regula. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 467–474, Taipei, May 2011. IFAAMAS.

[24] J. Mendling and M. Hafner. From WS-CDL choreography to BPEL process orchestration. *Journal of Enterprise Information Management*, 21(5):525–542, 2008.

[25] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative specification and verification of service choreographies. *ACM Transactions on the Web*, 4(1):3:1–3:62, 2010.

[26] OWL-S. OWL-S: Semantic markup for web services, Nov. 2004. http://www.w3.org/Submission/OWL-S/.

[27] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353, 2013.

[28] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[29] C. Peltz. Web service orchestration and choreography. *IEEE Computer*, 36(10):46–52, Oct. 2003.

[30] H. Purohit, Y. Ruan, D. Fuhry, S. Parthasarathy, and A. P. Sheth. On understanding the divergence of online social group discussion. In *Proceedings of the Eighth International Conference on Weblogs and Social Media*, 2014.

[31] D. Robertson and F. Giunchiglia. Programming the social computer. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.

[32] C. Rodríguez, F. Daniel, and F. Casati. Crowd-based mining of reusable process model patterns. In *Proceedings of the 12th International Conference on Business Process Management*, volume 8659 of *LNCS*, pages 51–66. Springer, 2014.

[33] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, pages 851–860. 2010.

[34] M. P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, Dec. 1998.

[35] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

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

[37] M. P. Singh. Cybersecurity as an application domain for multiagent systems. In *Proceedings of the 14th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1207–1212, Istanbul, May 2015. IFAAMAS. Blue Sky Ideas Track.

[38] M. P. Singh, M. Arrott, T. Balke, A. K. Chopra, R. Christiaanse, S. Cranefield, F. Dignum, D. Eynard, E. Farcas, N. Fornara, F. Gandon, G. Governatori, H. K. Dam, J. Hulstijn, I. Krüger, H.-P. Lam, M. Meisinger, P. Noriega, B. T. R. Savarimuthu, K. Tadanki, H. Verhagen, and S. Villata. The uses of norms. In G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre, editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, chapter 7, pages 191–229. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[39] M. P. Singh, A. K. Chopra, N. Desai, and A. U. Mallya. Protocols for processes: Programming in the large for open systems. *ACM SIGPLAN Notices*, 39(12):73–83, December 2004.

[40] L. A. Suchman. Office procedure as practical action: Models of work and system design. *ACM Transactions on Office Information Systems*, 1(4):320–328, Oct. 1983.

[41] A. H. Tapia, K. Bajpai, B. J. Jansen, and J. Yen. Seeking the trustworthy tweet: Can microblogged data fit the information needs of disaster response and humanitarian relief organizations. In *Proceedings of the 8th International Information Systems for Crisis Response and Management Conference*, pages 1–10, 2011.

[42] P. R. Telang, A. K. Kalia, J. F. Madden, and M. P. Singh. Combining practical and dialectical commitments for service engagements. In *Proceedings of the 13th International Conference on Service-Oriented Computing (ICSOC)*, volume 9435 of *Lecture Notes in Computer Science*, pages 3–18, Goa, India, Nov. 2015. Springer.

[43] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, Chichester, UK, 2009.

[44] WS-CDL. Web services choreography description language version 1.0, Nov. 2005. www.w3.org/TR/ws-cdl-10/.

[45] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 527–534. ACM Press, 2002.

[46] E. S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235. IEEE Computer Society, 1997.

[47] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering Methodology*, 12(3):317–370, 2003.

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