The AThOS Project: First Steps towards Computational Accountability

Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio

Università degli Studi di Torino — Dipartimento di Informatica
firstname.lastname@unito.it

Abstract. This work studies the support research on multiagent systems can give to the automation of computation of accountability (computational accountability), an din particular to accountability in organizational contexts. It introduces the idea of guaranteeing accountability as a design property of socio-technical systems, and explain a set of principles which a socio-technical system should respect. We explain the ADOPT protocol, which is a way that allows to realize computational accountability in an organizational setting, by means of multiagent systems technology.

Keywords: Computational Accountability, Business Artifacts, Normative MAS, Social Commitments

1 The AThOS Project: Motivations and Introduction

Thanks to the recent technological advances, that make devices such as smart phones and tablets indispensable tools in everyday life, the paradigm of Sociotechnical Systems (STSs) is rapidly spreading. STSs are complex systems where human beings interact with each other by means of ICT infrastructures. Humans may interact as members of the same organization, or enterprise, to achieve an organizational (business) goal. More often than not, however, interacting actors can belong to different organizations, and interaction is a means for reaching their goals.

In STSs, humans are the central elements of the system, they are essential to deliver the business goals. This central role of people makes STSs nondeterministic systems, because human behavior is generally unpredictable[1]. However, within organizations (and STSs) humans have relationships, duties, and commitments, whose satisfaction is crucial to the delivery of the business goals and, thus, to the well-being of the organization itself. Consistently, it is important, both to the employee and to the management, to be aware when duties (or commitments) are neglected, to understand the reasons, and decide if and how to act. The problem is that the mere availability of an ICT infrastructure does not support the interacting principals in being aware of the expectations that are put on

[1] In the sense that the same person may react in different ways to similar situations, depending on conditions that more often than not are not modeled in the system.

them, as it does not support who is in charge to get a clear picture of how tasks are, or are not, being carried on. This aspect becomes aggravated when the STS involves principals who belong to different organizations and *cross-organizational* relationships have to be established and maintained.

It is precisely because of the principals' autonomy that *accountability becomes a critical feature for an enterprise*. When something deviates from the expected and desired behavior, an accountable enterprise is committed to understand what went wrong and why, and to find a proper course of repairing action. What one expects from STSs is that such a process be, at least partially, automated. In other words, STSs should provide the ground for a form of *computational accountability* [4], where accountability is a property that can be checked at runtime, or even enforced at design time.

There are, indeed, attempts in business software applications to model the distribution of tasks and the related responsibility. For instance, the BPMN language for modeling workflows supplies the lane construct to distribute tasks among different actors. The idea is that an actor is responsible for performing the activities that lay in its lane, but a lot is left to intuition. Specifically, responsibility is not modeled explicitly, it is grasped from the workflow structure and may concern an office, not necessarily individuals. Then, being responsible for carrying out some activity does not mean being accountable for it. A more explicit instrument is the RACI matrix [29] by which each activity is associated with at least one responsible actor, exactly one accountable actor, and possibly some consulted and informed actors. The limit is that a RACI matrix is substantially an off-line resource that does not support actively an actor during her activities. It cannot be used as a monitoring tool on-line, but just as piece of data that a forum consults *a posteriori* to determine who was in charge of what. In [14] RACI matrices are automatically derived from BPMN specifications, but they are just used to support the resource assignment to each activity.

This paper reports the results of the two-year piloting phase of the *AThOS*[2] project. The ambitious, long-term objective of the AThOS project is the design of a framework for computational accountability in the context of cross-organizational business processes. From a methodological point of view, AThOS aims at providing a modeling framework for STSs, where the focus is shifted from a procedural representation of workflows to a *declarative* representation that is centered around the key notions of *interaction* and *social relationships* among actors. From a practical perspective, AThOS aims at providing a new enterprise software engine that monitors on-going interactions (and processes), and supports accountability. To achieve these results, the framework we are developing takes advantage of previous declarative approaches to the modeling of business processes, and integrates them with an explicit, first-class representation of the notion of "relationship", that relies on *social commitments* [27]. On the practical side, relying on previous works in literature (among which [28,13]), we are investigating the adoption of the paradigm of Multi-Agent Oriented Program-

[2] The *Accountable Trustworthy Organizations and Systems (AThOS)* project, funded by Università degli Studi di Torino and Compagnia di San Paolo (CSP 2014).

ming for the realization of business processes. This would allow a programmer to fully exploit the power of declarative agent programming frameworks (such as JaCaMo [10]).

The paper begins by explaining the kind of accountability the research focuses on (Section 2). Then, it introduces the notion of accountability by design, studying its feasibility in computational settings (Section 3). It continues by explaining the ADOPT protocol, which is a way to achieve computational accountability in multiagent systems (Section 4), and a first direction of implementation (Section 5). A discussion ends the paper (Section 6).

2 Organizational accountability and why it is important

Accountability is a wide-ranging concept, that can take on many different characteristics and meanings, depending on the discipline in which discussion takes place. So, the first point that it is relevant to underline is that our focus is posed on *accountability in organizational settings*. We do not care whether the organization is persistent, meaning that it has long-lasting goals that will produce many interactions (like a selling company), or if it is specifically created to achieve a once-in-a-time goal (like building a house), and will dissolve afterwards. We assume, however, the organization's functioning to be supported by an STS. In such a context, we aim at realizing *accountability as computational process*; more specifically, a backward-looking, institutional process that permits one entity to be held to account by another.

The accountability process activates when a state of interest is reached, and an *authoritative entity* wishes to hold to account those behind said state. Following [11], the process encompasses three primary phases: 1) an investigative entity (forum) receives all information regarding the agents' actions, effects, permissions, obligations, etc. that led to the situation under scrutiny, 2) the forum contextualizes actions to understand their adequacy and legitimacy, and finally 3) the forum passes judgment on agents with sanctions or rewards. Our goal consists in automating the entire process for use in a MAS, although we will presently leave out the sanctioning piece of the third phase, which is already studied in the literature on electronic institutions [16]. To realize our goal, we begin by looking at the reasoning process behind accountability to identify points of adaptation into the software world. Following [12], we believe that three conditions should all realize for an STS to support an accountability system: *agency*, *causal relevancy*, and *avoidance opportunity*. Such conditions enable accountability attribution. Agency means that the involved individuals are autonomous (indeed, in our setting they are autonomous by assumption), that they have reasoning capacity (idem), and that they can distinguish right and wrong. The third condition is the one that, in our understanding, calls for support by an STS. In order to distinguish right from wrong, individuals must be *aware* of the binds they have with the others and with the organization, as agent's "ethics" will be expressed through adherence to norms and commitments contextualized within an organization. Causal relevancy expresses the necessity of causation

linking a given agent to a situation under scrutiny. The avoidance opportunity condition specifies that an "agent should have had a reasonable opportunity to have done otherwise" [12]. Once again, awareness of the conditions an individual will be held to account for is crucial, since such knowledge has an impact on the choices of individuals as well as on its liability.

The realization of organizational accountability brings along practical advantages which motivate the purpose of studying accountablity as a system property. Evidence is provided by many studies and documents which promote the adoption of accountability frameworks. Among them, reports by the United Nations [33] and organizations like the OECD [26]. We briefly summarize the main reasons for realizing accountability frameworks and, thus, to study computational accountability. First of all, accountability within an organization is more than reporting information, disclosing reasons, or sanctioning the culprits. It is generally considered as a powerful feedback mechanism aimed at improving performance. An accountability framework, like the one of UNICEF [31] or WHO [32], explicitly establishes desired results, monitors and measures performance, uses the feedback obtained by the accountable parties to decide which changes should be made (which actions should be taken) to achieve the intended result. Such a process enables the evolution of an organization in a way that is functional to the fulfillment of its objectives. It is helpful both in those situations where the objectives, to be achieved, need to rely on external actors after the stipulation of a covenant, and in those cases where objectives depend entirely on processes that are internal to the organization itself. Another positive effect of organizational accountability is that it helps turning implicit expectations into explicit expectations. This is important because expectations that are not communicated create confusion and tensions inside an organization. Parties may be unaware of what they should deliver. Instead, a clear awareness of the expectations (and of the accountabilities) increases reliability, improves performance and trust, and helps to achieve common goals.

3 Accountability by Design

The research question that raises at this point is whether it is possible to create STSs that show *accountability as a design property*, that is, such that in every state the accountable parties are clearly identifiable. Providing accountability by design requires decision of how to tackle certain situations, long discussed in philosophy as ethical dilemmas. One of these concerns is causal determinism, that in our context we can rephrase as whether or not an agent should be considered accountable for an adverse state if this state is inevitable, whatever action the agent will perform. We adopt the incompatibilist position to moral responsibility and conclude in a software setting that an agent cannot be held accountable in causal determinism but that the discovery of inevitable states lies with the agents. If an agent stipulates provisions for an inevitable adverse state, that same adversity would be represented to some degree in its provisions due to its inevitable nature. The agent would still be held accountable for the

state, because it effectively declares responsibility for a goal through its accepted stipulated conditions. Likewise if an agent offers to realize a goal in the presence of a certain provisions, it is declaring for all intents and purposes that the goal is possible under certain conditions. Should that agent offer up an incorrect assessment and the goal be effectively impossible even with stipulations, the agent will nevertheless be held accountable, because the organization "believes" an agent's declaration. We therefore conclude, thanks to the distributed nature of our vision of accountability, that an organization can consider absent both impossibilities and inevitabilities. Another ethical dilemma concerns knowledge of the consequences of one's actions. That is, can one be held accountable for an unforeseeable outcome?

As an example, consider a setting where a house is being built. The companies and the workers who contribute to the construction constitute an organization. In particular, the organization involves two members: one who should prepare a wall, *wall-preparer*, and another, *painter*, who should paint the wall white at some agreed price. The *wall-preparer* fulfills her/his task by spackling the wall but instead of using a standard white spackle some dark colored one (a remainder of a previous work) is used. Due to this unexpected action, *painter* has not the correct amount of materials and cannot fulfill the painting task at the agreed price. Though not coerced, *painter* cannot fulfill what expected of him/her, due to the choices of another. Clearly, despite the fact that *wall-preparer* did what assigned, the kind of spackle that was used hampers the *painter*'s work. In this case it is difficult to identify an accountable party: the *painter* might have provided adequate provisions instead of giving them as granted, the *wall-preparer* might have asked if an unusual choice would have had consequences before doing the work, the organization might have checked that the two members properly coordinated. Their selfishness prevents their understanding the assigned tasks as part of a greater work, and the lack of an explicit account of the relationships between the agents (and their tasks) contributes to the picture but. On the other hand, if expectations are not clear and shared, how could it be otherwise? For instance, *wall-preparer* does not know the circumstances under which the *painter* can achieve its goal, should it be held accountable for its decision of using a colored spackle? To foster the good functioning of the organization, then, there is the need of adequate support.

3.1 Characterization of Organizational Accountability

Accountability as a design property means that an STS is built in such a way that accountability can be determined from any future institutional state. Indeed, our goal lies in automating the entire process, that is, to create a structure that creates and collects contextualized information so that accountability can actually be determined from any future institutional state. We consider integral to this process the following steps: a forum must receive all information, including all causal actions, regarding a given situation under scrutiny, the forum must be able to contextualize actions to understand their adequacy and legitimacy, and finally the forum must be able to pass judgment on agents. One of the key

difficulties in realizing our goal lies with the notion of *contextualized action*. In our own societies, contextualizing might entail an examination of circumstances: for example, what should have a person done, why didn't she/he do that, what impact did her/his actions have, and given what the person had to work with, did she/he act in an exemplary fashion? The same process in a MAS would be guided by the same type of questions, though in order to facilitate the answers, we need to make use of different structures. In particular, we need structures that allow assessing who is accountable without actually infringing on the individual and private nature of agents.

We identify the following necessary-but-not-sufficient principles a MAS must exhibit in order to support the determination of organizational accountability.

Principle 1 All collaborations and communications subject to considerations of accountability among the agents occur within a single scope that we call *organization*.

Principle 2 An agent can enroll in an organization only by playing a *role* that is defined inside the organization.

Principle 3 An agent willing to play a role in an organization must be aware of all the powers associated with such a role before adopting it.

Principle 4 An agent is only accountable, towards the organization or another agent, for those goals it has explicitly accepted to bring about.

Principle 5 An agent must have the leeway for putting before the organization the provisions it needs for achieving the goal to which it is committing. The organization has the capability of reasoning about the requested provisions and can accept or reject them.

Principle 1 calls for situatedness. Accountability must operate in a specific context because individual actions take on their significance only in the presence of the larger whole. What constitutes a highly objectionable action in one context could instead be worthy of praise in another. Correspondingly, a forum can only operate in context and an agent's actions must always be contextualized. The same role in different contexts can have radically diverse impacts on the organization and consequently on accountability attribution. When determining attribution, thus, an organization will only take into account interactions that took place inside its boundaries. Placing an organizational limit on accountability determination serves multiple purposes. It isolates events and actors so that when searching for causes/effects, one need not consider all actions from the beginning of time nor actions from other organizations. Agents are reassured that only for actions within an organization will they potentially be held accountable. Actions, thanks to agent roles (Principle 2), also always happen in context.

To adequately tackle accountability by categorizing action, we must deal with two properties within a given organization: 1) an agent properly completes its tasks and 2) an agent does not interfere with the tasks of others. The principles 2–5 deal more explicitly with the first property, i.e., how to ensure that agents complete their tasks in a manner fair for both the agents and the organization. The second property is also partially satisfied by ensuring that, in the presence of goal dependencies, the first agent in sequence not to complete its goal will bear

accountability, not only for its incomplete goal, but for all dependent goals that will consequently remain incomplete. That is, should an agent be responsible for a goal on whose completion other agents wait, and should that agent not complete its goal, then it will be accountable for its incomplete goal and for that goal's dependents as well.

As an organizational and contextual aid to accountability, roles attribute social significance to an agent's actions and can, therefore, provide a guide to the severity of non-adherence. Following the tradition initiated by Hohfeld [20], a power is "one's affirmative 'control' over a given legal relation as against another." The relationship between powers and roles has long been studied in fields like social theory, artificial intelligence, and law. By Principle 3 we stipulate that an agent can only be accountable for exercising the powers that are publicly given to it by the roles it plays. Such powers are, indeed, the means through which agents affect their organizational setting. An agent cannot be held accountable for unknown effects of its actions but, rather, only for consequences related to an agent's known place in sequences of goals. On the other hand, an agent cannot be held accountable for an unknown goal that the organization attaches to its role, and this leads us to Principle 4. An organization may not obligate agents to complete goals without prior agreement. In other words, an organization must always communicate to each agent the goals it would like the agent to pursue. Notice that with this principle we diverge from considerations in the field of ethics regarding accountability in the presence of causal determinism [19,12], where even in the absence of alternate possibilities humans can be morally responsible thanks to the significance of the choice to act. Finding the conversation fundamentally shifts when speaking of software agents, we consequently conclude that accountability is not attributable in the presence of impossibilities. Correspondingly, agents must be able to stipulate the *conditions* under which a given goal's achievement becomes possible, i.e. the agent's requested *provisions*. The burden of discovery for impossibilities, therefore, rests upon an agent collective who announce them by their combined silence for a given goal. That is, a goal becomes effectively impossible for a group of agents should no agent stipulate a method of achievement. Conversely, an agent also declares a goal possible the moment it provides provisions to that goal. Should an uniformed agent stipulate insufficient provisions for an impossible goal that is then accepted by an organization, that agent will be held accountable because by voicing its provisions, it declared an impossible goal possible. The opportunity to specify provisions, therefore, is fundamental in differentiating between impossibilities and possibilities.

Going back to the painter example, before agreeing to be the organization's painter, *painter* would stipulate provisions for its role goals, in this case, *white wall*. An organization, accepting *painter*'s provisions, would then add *white wall* as a goal condition to the job description of *wall-preparer*. *Wall-preparer* would in turn accept its new goal condition. Come execution time, when *wall-preparer* adds the black stripe, not only will *painter*'s provisions not be met, but *wall-preparer* will also have violated its own goal conditions. Since *wall-preparer*

knows what it did was wrong thanks to goal conditions, causally contributed to an adverse situation of work failure, and could have avoided causally contributing, it will be held accountable for the adverse state.

4 Reaching Accountability via the ADOPT Protocol

We now introduce our vision to achieve computational accountability via a proper design of the interactions between software components. The first step is a paradigm shift: from the *process-oriented* view to the *agent-oriented* view. The process-oriented view is substantially activity-centric – that is, procedural. It focuses on the (business) process that is carried on by a software component, but overlooks the interaction between components, at the core of the realization of STSs. Instead, an STS can be conveniently thought of as a Multi-agent System (MAS) where software agents interact with each other on behalf of human principals. An advantage of the agent perspective is that it enables the use of some design (and programming) abstractions.These are, for instance, the *environment* where the agents operate, the *norms* governing *organizations* and *institutions*, the *roles*, and their *powers*, possibly defined therein, and an explicit representation of interactions, for instance, via social *commitments*.

The agent abstraction allows us to think of software modules as goal-oriented components. That is, the focus moves from the activities within a process to the *goal* the process aims at reaching. The design process can therefore be modularized, because the interactions among agents (i.e., components) can be kept separated from the inner process of each agent. In fact, interactions can be specified as a consequence of the agents' goals. That is, the agents create and maintain interactions as a means for reaching their goals. The internal process of the agent lies at a different design level with respect to interaction. For the accountability purpose, only the interaction level is relevant since it makes explicit the engagements that agents have agreed upon. Monitoring the progress of these engagements is one the main objectives of an accountable system.

Our intuition is that an STS can be seen as an *organization* in which agents (i.e., the software components of the STS at hand) can join and operate. More precisely, we follow the ontological definition of organizations given in [9]. Organizations are characterized by *roles*, which are first-class modeling entities. Roles are actually social roles, and are related to their players (i.e., the agents), and their organization by three properties: (1) a role must always be associated with the institution it belongs to and with its player; (2) a role is meaningful only within the institution it belongs to, and hence the definition of a role is only given within an institution; (3) the actions defined for a role in an institution have access to the state of the institution and of other roles; these actions are, therefore, the powers a role player is endowed with. In order to obtain an accountable organization, we require that the interactions between the organization and its members follow a given *accountability protocol* (ADOPT[3] [3,5]) that,

[3] Accountability-Driven Organization Programming Technique

by realizing the five principles introduced above, guarantees accountability as a design property of the system as a whole. In essence, the accountability protocol makes the legal relationships between each agent and its organization explicit. Relationships are expressed as a set of (abstract) commitments (i.e., contracts), directed from the agents towards the organization itself, and vice versa. ADOPT consists of two phases: the *enactment* phase, and the *goal-assignment* phase.

Enactment Phase. In this phase, an agent enrolls in an organization by playing a role. During this phase, the agent is made aware of the powers the organization will grant to it as a player of a specific role, and the agent will commit towards the organization to use these powers when time will demand it. Let Ag_i be the agent willing to enroll in organization Org:

(1) Ag_i : commit to be a player for role R_i with powers $pwr_{i,1} \dots pwr_{i,n}$ if Org accepts
(2) Org : accept Ag_i as player of role R_i
(3) Ag_i : commit to use the powers $pwr_{i,1} \dots pwr_{i_n}$ when Org demands to

In step (1) agent Ag_i asks Org to join the organization becoming a player of role R_i, with powers $pwr_{i,1} \dots pwr_{i,n}$. Note that a power calls for a capability of the agent to exercise it. For instance, the *wall-preparer* is given the power to prepare the walls of the house being built, but to prepare the wall it needs materials, skill, and to take approriate action. Thus, the powers associated with role R_i will pose some capability requirements on the agent side. An agent willing to play role R_i is made aware of the capabilities it has to possess for being a proper actor of that role. The powers will be the means through which the agent will operate within the organization by changing the state of the organization itself. Capabilities, instead, will be sets of behaviors, owned by the agent, that allow it to exercise the powers that go with some role the agent plays. In the enactment phase the agent takes on the responsibility of this (previously implicit) declaration. This allows overcoming the impossibility to inspect the agents' code (which is not disclosed) that characterizes, in particular, cross-organizational settings as well as many STSs: the organization cannot verify whether the agent's implementation is compliant with the capability requirements, but if, during the execution, the agent will be unable to exercise one of its powers, *it will be deemed accountable* thanks to the commitments established during this phase.

The internal decision process, by which Org decides whether to accept the agent, are outside the scope of ADOPT. Here we are only interested in the final decision of accepting the agent. Step (2) encodes such a case. Once Org has accepted Ag_i, this agent is obliged, by the engagement created at step (1), to commit to use R_i powers in case Org will ever need them. This is done with the commitments created at step (3). After these three steps, agent Ag_i is enrolled within the organization. Notably, these few steps are sufficient to satisfy the first three principles we have identified. In fact, the protocol imposes the existence of an organization within which all the relevant events must occur (Principle 1). The protocol allows an agent to be part of an organization only by playing a role within that organization (Principle 2), and finally, the protocol is such that an agent is aware of the powers it will possibly use within the organization as player of a specific role (Protocol 3).

Goal-assignment Phase. The second phase of ADOPT considers the assignment of goals to some member of the organization. According to Principle 5, an agent should be able to negotiate with the organization the provisions it needs in order to accomplish a given goal. This negotiation is not part of the ADOPT proposal, and any negotiation algorithm can be used. Here, we just assume that after the negotiation, the agent and the organization have agreed upon the goal $g_{i,k}$ the organization wants Ag_i to perform, and the provision $prov_{i,k}$ the agent demands to the organization as a prerequisite. Such an agreement is "sealed" by the agent and the organization in the ADOPT protocol as below outlined.

(4) Org : commit to assign goal $g_{i,k}$ with provisions $prov_{i,k}$ if Ag_i accepts
(5) Ag_i : commit to bring about $g_{i,k}$ if Org assigns it and brings about provisions $prov_{i,k}$
(6) Org : assign $g_{i,k}$ to Ag_i
(7) Org : bring about provision $\text{prov}_{i,k}$
(8) Ag_i : achieve goal$g_{i,k}$

In step (4), Org commits towards agent Ag_i that, if the agent will accept to accomplish the goal, it will assign the goal to the agent, and will supply Ag_i the agreed provisions. Thus, also Org takes on commitments towards its members. These commitments, in particular, assure that Org will never ask an agent to bring about a goal, for which either no agreement was reached, or provisions are missing. In case Org contravenes this expectation, it will be held accountable as any other player in the system. In step (5), Ag_i creates the commitment of bringing about goal $g_{i,k}$ provided that the provisions $prov_{i,k}$ have been supplied, and the goal has been assigned to it. In the last three steps, the two parties operate so as to satisfy their commitments. Org will bring about the provisions[4], and will assign goal $g_{i,k}$ to Ag_i. On its side, Ag_i will try achieve the assigned goal so as to satisfy its commitment towards Org.

ADOPT supports the realization of an accountable system through the mutual commitments that exist between Org and any agent Ag_i, which are made explicit. This feature realizes Principle 4, for which an agent is accountable only for those goals it has explicitly accepted to achieve. In fact, in our proposal, an organization cannot command one of its members to achieve a goal. Rather, the organization will propose a goal, and the agent will have to decide whether explicitly accepting that goal. Only in this way it becomes possible to assess accountability. It is possible to verify that the ADOPT protocol satisfies Principle 4 via model checking. In fact, differently from the other principles that are directly satisfied by the structure of the organization we propose, the verification of this principle demands consideration of the possible evolutions of the protocol. Indeed, one has to guarantee that, in any possible run of the protocol, the organization assigns a goal to an agent only when this goal is accepted. In [3] we discuss how the ADOPT protocol can be encoded as an interpreted system [18], and then, by means the MCMAS model checker [21], the validity of Principle 4 can be verified in terms of CTL formulae.

[4] Provisions will reasonably be supplied by other agents within the organization.

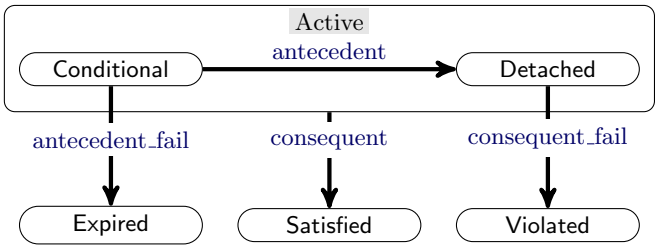

Fig. 1. Commitment life cycle.

5 Implementing ADOPT and Future Directions

One straightforward way for implementing ADOPT is to rely on the notion of *social commitment* [15,27]. A social commitment (sometimes called commitment, for simplicity, in the following) is a sort of contract between two parties: a *debtor* commits towards a *creditor* to bring about a consequent condition *con* should an antecedent condition *ant* occur. Social commitments, thus, find a direct mapping with the commitments that arise in the ADOPT protocol. Formally, commitments have the form $\mathsf{C}(debtor, creditor, ant, con)$, and their evolution follows the lifecycle reported in Figure 1: a commitment is *Violated* either when its antecedent is true but its consequent will forever be false, or when it is canceled when Detached. It is *Satisfied*, when the engagement is accomplished (notice that a commitment does not need to be detached before being satisfied). It is *Expired*, when it is no longer in effect and therefore the debtor would not fail to comply even if does not accomplish the consequent. Active has two substates: *Conditional* as long as the antecedent does not occur, and *Detached* when the antecedent has occurred. Commitments have a normative value because when the antecedent condition is satisfied, the debtor is obliged to bring about the consequent, lest the violation of the commitment. Differently from obligations in deontic logic, however, a commitment can only be created by its debtor; thus, its creation is a consequence of an internal process of the debtor, rather than the consequence of a normative system as it happens with obligations.

Commitments can be used to specify the semantics of message-based protocols, or they can be a means for specifying protocols on their own, see for instance the JaCaMo+ agent platform [1], an extension of JaCaMo [10], where commitment-based protocols are made available as a specification of protocol artifacts. Commitments are, therefore, a viable tool for the realization of accountable systems because, on the one side, the ADOPT protocol is easily mapped into a commitment-based interaction protocol and, on the other side, there are agent platforms that support commitments as a native programming element, providing a technological infrastructure for the implementation of accountable STSs. The adoption of commitments opens also novel development directions, that we will shortly outline in the rest of this section.

5.1 Accountability of Information Management with Normative Business Artifacts

In the quest for the computational accountability, we have previously advocated a shift from a procedural view to an agent-oriented view, which usually features on a declarative approach. Think, for instance to BDI agents and the way they are realized in many agent-based platforms (e.g. JaCaMo [10] and JaCaMo+ [1]). Indeed, the use of declarative formalisms for Business Process Management is not new. BALSA [7] is a first attempt to define processes in terms of declarative ECA-rules (i.e., Event-Condition-Action). The BALSA methodology relies on the important notion of *business artifact* [25]. A business artifact is a key conceptual business entity that is used in guiding the operation of a business. It is a concrete, identifiable, self-describing chunk of information. It is characterized by two specifications: (1) an *information model* that defines what relevant data are traced by the artifact, and (2) a *lifecycle model* that defines how such data can evolve as a consequence of business operations on the artifact itself. Importantly, both specifications are accessible to the processes that use the business artifact. The BALSA methodology is targeted to specify a data-centric declarative model of business operations. It can be summarized in three steps: 1) identify the relevant business artifacts of the problem at hand and their lifecycles, 2) develop a detailed specification of the services (or tasks) that will cause the evolution of the business artifact lifecycles, 3) define a number of ECA-rules that create associations between services and business artifacts. ECA-rules thus are the building blocks to define, in a declarative way, processes operating on data.

BALSA is extremely interesting, in particular because it introduces a novel perspective on the modeling of business processes. However, when it comes to the coordination of different business processes, the methodology refers to choreography languages and techniques proposed for service-oriented computing. Thence one of the major disadvantages of this approach and of its descendants, that is the lack of a clear separation of concerns between the *coordination logic* and the *business logic*. A business process will internalize the message exchange specified by the choreography at hand. Should the choreography be changed, it would be necessary to modify also the business processes accordingly. The happens because choreographies realize a form of subject coordination.

In [2] we introduce the notion of *normative business artifact*, as a way to obtain objective coordination. Inspired by the *activity theory* [17], we propose to use a business artifact as a coordination medium. To this end, the business artifact is enhanced with a normative layer in which the mutual expectations that each agent has on the others are made explicit, and hence can be used for coordinating the agents' activities. Apart from the advantage of having a form of objective coordination, normative business artifacts allow one to think about accountability of data and information management. That is, instead of considering just the state changes of a piece of information, it is also important to take into account other aspects of information management such as its correctness, its accessibility and spreading (inside and outside an organization), its usage during the decision making of agents, and so on. An interesting direction

of research is to investigate the use of social commitments for tackling these information management aspects as *accountability requirements* that, specified at design time, become a guide for the practical development of accountable STS.

5.2 Practical Reasoning and Accountability

The ADOPT protocol aims at supporting accountability at the organizational level, that is, ADOPT formalizes how an accountable relationship can be established between an organization and each of its members. Our long-term goal, however, is to realize an accountable system where the accountability property cuts across all aspects of an organization, and hence it holds not only between the organization and each member, but also between any two members of the organization. This is particularly important, and challenging, in an STS, where agents are autonomous in the achievement of their goals, and the interactions raising in this process may be problematic. The intuition, thus, is that every interaction that arises between any two agents in an organization should be accountable, in the sense that it should always be possible to identify an agent that is held to account towards another agent about its conduct. It is worth noting, however, that the accountability property must not be thought of as a means for limiting the autonomy of the agents. On the contrary, in our view, an accountable system supports the agents in their practical reasoning by making their (mutual) expectations explicit.

Commitments can play a fundamental role in realizing this vision since they formalize a tie between the internal goal of an agent with the relationships that the same agent may create in order to achieve that goal. In [30] this relation between goals and commitments is formalized by an operational semantics given as a set of patterns of practical reasoning rules. The Social-Continual Planning (SCP) technique introduced in [6] is a first attempt to integrate practical rules within a planning framework. SCP enables agents to plan their interactions as a means for achieving their goals. As future work, we intend to extend the SCP technique so as to guarantee that the planned interactions compose, altogether, an accountable system.

Another important aspect that we have to consider in the development of an accountable system, is that agents are autonomous and can violate their commitments on purpose. This eventuality must not be overlooked by a well-designed accountable framework; nonetheless, an accountable system is not, in our view, finalized to sanctioning the agents that violate their commitments (although sanctioning mechanisms could be contemplated). Our goal, in fact, is in providing agents with feedback about their accountability relationships. Under this respect, an accountable framework should include some form of diagnostic reasoning (see e.g., [24,23]), in order to detect deviations from the expected evolutions of an interaction, and provide the involved agents with possible explanations of what has occurred. These explanations (the feedback) are an important source of information that the agents can exploit in their practical reasoning for recovering from the violation of some commitments. A plan repair technique, driven by the results of a diagnostic inference in a multi-agent setting, is discussed in [22]. In

the next future, we aim at integrating such a plan repair technique within SCP, this would allow agents not only to plan their interactions, but also to properly react when these interactions do not progress as expected.

6 Discussion and Conclusions

In this paper we have summarized the main results about computational accountability we have collected in the scope of the AThOS project. AThOS was a two-year, local project completed in 2017, whose main objective was to shed some light on the design and development of accountable Sociotechnical Systems (STSs). Current methodologies for the development of STSs, are mainly focused on *procedural* aspects, with the purpose of gaining in efficiency of execution. Many such systems rely on languages (like BPMN, YAML, or UML Activity Diagrams) and related tools for modeling workflows. The procedural perspective is inadequate for the computational accountability purpose. First of all, it results in a too rigid model where the possible lines of activity are all delineated a priori. Each actor is forced to follow a predefined course of action, without the possibility to deviate for taking advantage of arising opportunities or for managing unexpected emergencies. More importantly, a procedural perspective, being activity-centric, overlooks the interaction dimension, which is central for the accountability reasoning. An accountability system, in fact, should in principle involve at least two actors: one which is held to account for its actions, and one (often referred to as the *forum*) which judges the conduct of the previous one. Both actors rely on the relationships they have created, and on the mutual expectations these relationships induce, to sustain their position (i.e., good vs. bad conduct). The AThOS project answers to these deficiencies by modeling STSs based on the typical abstractions provided by the agent-oriented paradigm (e.g., electronic institutions, organizations, norms, etc.). By leveraging on these abstraction, the main contribution of AThOS lies in the definition of *organizational accountability*: the STS becomes the organization within which all the relevant interactions among the actors (i.e., the agents) occur, and are explicitly traced. We have introduced five principles underpinning the accountability at the organizational level, and presented the ADOPT protocol as a means for satisfying these principles at the design of the STS. From a practical point of view, we have envisaged the use of social commitment for representing, and manipulating, the accountability relationships between the organization and its members.

Social commitments pave the way to further extensions that are sketched in the paper. They provide a promising solution for dealing with accountability of data and information management, overcoming some limits of the artifact-centric proposals [7,8], namely the use of choreographies for the synchronization of the activities of different processes. Since choreographies capture interaction only in terms of exchanged messages, it will not be possible, by looking at a choreography, to reason on the relationships between some actors, or to have expectations on their behavior. In AThOS, we have investigated an extension to business artifacts with a normative layer. Such a normative layer makes ex-

plicit the mutual expectations that processes (i.e., agents in our agent-oriented perspective) may have on each other while using a given business artifact. A first advantage of normative business artifacts is that the coordination among agents needs not a further modeling element such a choreography. Indeed, the business artifact itself becomes the medium of coordination, with a significant improvements from the software engineering point view, as we have observed. In addition, normative business artifact could be the key for the definition of accountability of data and information. The idea, is not only to model how the information can evolve over an interaction, but also how operations on data influence the way in which agents behave.

Another interesting line of research that stems from using social commitments is about acting in an accountable system. In [6] we have introduced a methodology–Social Continual Planning–integrating practical reasoning patterns relating goals and commitments [30] into a multi-agent planning framework. The technique is a good starting point for developing systems where the accountability property is guaranteed not only between an agent and the organization it belongs to, but also between any two agents that come to interact in the attempt to achieve their own goals. Thus, this is a promising line for reaching the desiderata of establishing accountable relationships across all levels of an organization.

References

1. M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio. Commitment-based Agent Interaction in JaCaMo+. *Fundamenta Informaticae*, 157:1–33, 2018.
2. Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Objective Coordination with Business Artifacts and Social Engagements. In E. Teniente and M. Weidlich, editors, *Business Process Management Workshops, BPM 2017 International Workshops, Revised Papers*, volume 308 of *Lecture Notes in Business Information Processing (LNBIP)*, pages 1–18, Barcelona, Spain, 2018. Springer. This paper has been presented at the First Workshop on BP Innovations with Artificial Intelligence, BPAI 2017, organized by Koehler, J., Montali, M., Srivastava, B., Stuckenschmidt, H., De Masellis, R., Di Fracescomarino, C., Maggi, F. M., and Senderovich, A.
3. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. ADOPT JaCaMo: Accountability-Driven Organization Programming Technique for JaCaMo. In A. Bo, A. Bazzan, J. Leite, L. van der Torre, and S. Villata, editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems, 20th International Conference*, number 10621 in Lecture Notes in Computer Science, pages 295–312, Nice, France, October 30th–November 3rd 2017. Springer.
4. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Computational accountability. In *Proceedings of the AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents 2016*, volume 1802 of *CEUR Workshop Proceedings*, pages 56–62. CEUR-WS.org, 2017.
5. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Supporting Organizational Accountability inside Multiagent Systems. In R. Basili, F. Esposito, S. Ferilli, and F. A. Lisi, editors, *AI*IA 2017:*

Advances in Artificial Intelligence, XVI International Conference of the Italian Association for Artificial Intelligence, volume 10640 of *Lecture Notes in Computer Science*, pages 403–417, Bari, Italy, November 14th–17th 2017. Springer.
6. Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio. Social continual planning in open multiagent systems: A first study. In *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings*, pages 575–584, 2015.
7. Kamal Bhattacharya, Nathan S. Caswell, Santhosh Kumaran, Anil Nigam, and Frederick Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4):703–721, 2007.
8. Kamal Bhattacharya, Richard Hull, and Jianwen Su. *A data-centric design methodology for business processes*, pages 503–531. Handbook of Research on Business Process Modeling. IGI Publishing, 2009.
9. Guido Boella and Leendert W. N. van der Torre. The Ontological Properties of Social Roles in Multi-Agent Systems: Definitional Dependence, Powers and Roles playing Roles. *Artificial Intelligence and Law Journal (AILaw)*, 15(3):201–221, 2007.
10. Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Sci. Comput. Program.*, 78(6):747–761, 2013.
11. Mark Bovens, Robert E. Goodin, and Thomas Schillemans, editors. *The Oxford Handbook of Public Accountability*. Oxford University Press, 2014.
12. Matthew Braham and Martin van Hees. An anatomy of moral responsibility. *Mind*, 121(483), 2012.
13. P. Bresciani and P. Donzelli. A Practical Agent-Based Approach to Requirements Engineering for Socio-technical Systems. In *Agent-Oriented Information System*, volume 3030 of *LNCS*. Springer, 2003.
14. Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés. *Automated Resource Assignment in BPMN Models Using RACI Matrices*, pages 56–73. Springer, Berlin, Heidelberg, 2012.
15. Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *ICMAS*, pages 41–48. The MIT Press, 1995.
16. Mark d'Inverno, Michael Luck, Pablo Noriega, Juan A. Rodríguez-Aguilar, and Carles Sierra. Communicating open systems. *Artif. Intell.*, 186:38–94, 2012.
17. Yrjö Engeström, Reijo Miettinen, and Raija-Leena Punamäki, editors. *Perspectives on Activity Theory*. Cambridge University Press, Cambridge, UK, 1999.
18. Ronald Fagin, Joseph Y. Halpern, and Moshe Y. Vardi. *Reasoning about knowledge*. MIT Press, 1995.
19. Harry G. Frankfurt. Alternate possibilities and moral responsibility. *The Jounral of Philosophy*, 66(23), 1969.
20. Wesley Newcomb Hohfeld. Some fundamental legal conceptions as applied in judicial reasoning. *The Yale Law Journal*, 23(1):16–59, 1913.
21. Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.
22. Roberto Micalizio. Action failure recovery via model-based diagnosis and conformant planning. *Computational Intelligence*, 29(2):233–280, 2013.
23. Roberto Micalizio and Pietro Torasso. Agent cooperation for monitoring and diagnosing a MAP. In *Multiagent System Technologies, 7th German Conference, MATES 2009, Hamburg, Germany, September 9-11, 2009. Proceedings*, pages 66–78, 2009.

24. Roberto Micalizio and Pietro Torasso. Cooperative monitoring to diagnose multiagent plans. *Journal of Artificial Intelligence Research*, 51:1–70, 2014.
25. Anil Nigam and Nathan S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428 – 445, 2003.
26. OECD. Effective institutions and the 2030 agenda for sustainable development. http://www.effectiveinstitutions.org/media/Session_1_Background_note_SDG.pdf.
27. Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
28. Munindar P. Singh. Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology*, 5(1):21:1–21:23, 2013.
29. Michael L. Smith and James Erwin. Role and responsibility charting (RACI). https://pmicie.starchapter.com/images/downloads/raci_r_web3_1.pdf, 2005.
30. Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith. Relating Goal and Commitment Semantics. In *Post-proc. of ProMAS*, volume 7217 of *LNCS*. Springer, 2011.
31. United Nations Children's Fund. Report on the accountability system of UNICEF. https://www.unicef.org/about/execboard/files/09\discretionary{-}{}{}15\discretionary{-}{}{}accountability\discretionary{-}{}{}ODS-English.pdf, 2009. E/ICEF/2009/15.
32. World Health Organization. WHO accountability framework. http://www.who.int/about/who_reform/managerial/accountability-framework.pdf, 2015.
33. Mounir Zahran. Accountability Frameworks in the United Nations System. https://www.unjiu.org/en/reports-notes/JIU%20Products/JIU_REP_2011_5_English.pdf, 2011. UN Report.