Fragility and Robustness in Multiagent Systems

Matteo Baldoni, Cristina Baroglio(✉), and Roberto Micalizio

Dipartimento di Informatica, Università degli Studi di Torino, Turin, Italy
{matteo.baldoni,cristina.baroglio,roberto.micalizio}@unito.it

Abstract. Robustness is an important property of software systems, and the availability of proper feedback is seen as crucial to obtain it, especially in the case of systems of distributed and interconnected components. Multiagent Systems (MAS) are valuable for conceptualizing and implementing distributed systems, but the current design methodologies for MAS fall short in addressing robustness in a systematic way at design time. In this paper we outline our vision of how robustness in MAS can be granted as a design property. To this end, we exploit the notion of accountability as a mechanism to build reporting frameworks and, then, we describe how robustness is gained. We exemplify our vision on the JaCaMo agent platform.

Keywords: Robustness · MAS engineering · Accountability · JaCaMo

1 Introduction

Robustness is an important property of software systems. The Systems and Software Engineering Vocabulary ISO/IEC/IEEE 24765 international standard defines it as the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions [1]. In many cases, robustness refers to a system property rather than to the system as a whole: a property of a system is robust if it is invariant with respect to a set of perturbations [2]. This makes it possible to interpret many system properties as types of robustness: reliability as robustness to component failures; efficiency as robustness to lack of resources; scalability as robustness to changes to the size and complexity of the system as a whole; modularity as robustness to structured component rearrangements; evolvability as robustness of lineages to changes on long time scales.

The availability of feedback is seen as crucial in gaining robustness [2], yet not easy to obtain as is the case of multi-scale systems or of distributed systems of interconnected components. We see feedback as a piece of information, broadly speaking some facts that are obtained retroactively, that objectively concern an execution of interest, and that are passed from one component to another. The significance and the quality of feedback are crucial, as well, in making a system robust: one would not want any kind of information to be returned but only information that is functional to the desired kind of robustness, and that comes from a reliable source.

Coming to Multiagent Systems (MAS), the current architectures and methodologies for their design and development (see e.g., [3,44,47]) fall short in addressing robustness in a systematic way at design time. For instance, they do not foresee mechanisms for exception handling, as instead is done for programming languages like Java, or in the actor model (e.g., see [30]). This happens because traditional approaches to exception handling do not accommodate some important features of MAS, like openness, heterogeneity, agent encapsulation, and distribution [37]. The common assumption is that software components are "collaborative", and that the code will be available for inspection. But introspection is often impossible when dealing with agents, and collaboration cannot be given for granted. As maintained in [37], in the case of MAS this sort of mechanisms should leverage on the proactivity of the agents.

Nevertheless, it is already possible to see in action elements that support the systematic introduction of robustness. For instance, agents often rely on reputation and trust to estimate how reliable another agent is before using (in their deliberative cycle) a piece of information that was produced by that agent [33]. When the MAS is enriched with an organizational infrastructure, that same infrastructure can be exploited to state the authority of the agents on given scopes. However, in order to support robustness something more is still needed. Alderson and Doyle [2] suggest that a possible strategy to achieve robustness in complex systems consists in "using feedback interconnection of sensors and actuators". That is, by exploiting the feedback coming from a (network of) system sensor(s), a component (in our case, an agent) can properly activate its actuators to complete its task. In this paper we aim at mapping this vision in the context of Multiagent Organizations (MAO). In MAO, is our opinion, feedback and feedback networks must be encompassed at the institutional level, through mechanisms that, on the one hand, are seamlessly integrated in the organizational ones (basically, the use of norms to regulate the functioning of the organization) and, on the other hand, capture the interconnectedness of feedback production and its propagation.

In the next section we explain the difficulties of gaining robustness in MAS. In Sect. 3 we introduce accountability and lay the basics for building reporting frameworks through it; in Sect. 3.1 we introduce a possible architecture, extending that of JaCaMo [13], for robust MAO.

2 Fragility in Distributed Systems and MAS

Many systems "are complex networks of multiple algorithms, control loops, sensors, and human roles that interact over different time scales and changing conditions" [43]. In sociology, such a complex network becomes a set of constraints that make a system, which comprises many parts, to act as a whole [24]. The combination of individuals and relationships produces emergent powers that enable the organization to achieve goals that otherwise would not be achievable (or not as easily). And the same holds for MAO.

However, the greater complexity introduces also new fragilities, that need to be coped with. More generally, "... this complexity itself can be a source of new fragility, leading to 'robust yet fragile' tradeoffs in system design" [2]. For example, consider the autonomous vehicle described in [43]. It is equipped with eighteen sensor packages, basic sensor processing/actuator controls, software or reasoning on temporal logic, sensor fusion, multiple path, traffic, and mission planners, conflict management, health monitoring, fault management, optimization, classifiers, models of the environment (maps), obstacle detection, road finding, vehicle finding, and sensor validation checks. Here, the use of protocols, of layering, and of feedback creates a complex, multi-scale modularity that *per se* is exposed to many risks of failure in presence of abnormal conditions. How to gain robustness?

It is possible to resort to MAS abstractions and methodologies to tackle the realization of robust complex systems of the described kind. These methodologies typically assume that agents coalesce in *organizations* to coordinate their interactions and tasks: system-level goals can be accomplished by taking advantage of the contribution of each agent [41]. An organization, thus, encompasses a functional decomposition of a global goal into subgoals. Subgoals are, then, assigned to agents by means of *norms*, that orchestrate the execution of the functional decomposition: as soon as a specific organizational goal is needed, the normative system generates an *obligation* toward some agent to achieve that goal. Agents' acceptance of the organizational constraints enables them to act in a shared environment, and achieve results unachievable if they acted in isolation. The agents' autonomy is an enabler of the system adaptability, which, in turn, is crucial to achieve robustness: a robust system is one that adapts to stressful environmental conditions, and components can adapt to changing contextual conditions and perturbations only if they are autonomous in their decision process. Adaptability, however, requires the system to be equipped with the ability to produce proper feedback, propagate it, and process it, so as to enable the selection and enactment of behavior that is appropriate to cope with the situation. The lack of such mechanisms makes the system fragile.

A functional decomposition describes what is expected of the agents for achieving a global goal, based on their supposed capabilities, but agents may fail the expectations. When this happens, a normative system would typically take the involved agents as *violators of some obligation*, and react to the violation by issuing sanctions toward the misbehaving agent. Consequently, on one hand, we can see the normative system as a means that enables the orchestration of the activities of a group of autonomous agents, while on the other hand, in some sense we can see it also as a means that tries to produce robustness. This because the agents are pushed to do *what is expected of them*, and thus to tackle the situations the system is facing. The rationale is to guide the agents toward the interest of the organization. The problem is that sanctions are not generally accompanied by feedback and feedback handling mechanisms, and thus they do not provide a means that supports robustness. Indeed, to be effective, sanctions must at least (1) be sufficiently "strong" to contrast the agents' self-interest in pursuing different goals of their own, and (2) target agents that actually have the resources and capabilities needed to face the situation of interest. In both cases robustness would be gained only by propagating through the system information about the *reasons* that caused the violation, and by revising the norms accordingly. Otherwise, for what concerns the first condition, how to identify a right trade-off that works for any agent without making assumptions of the agents' internals? For what concerns the second condition, how to propagate the reasons that cause the failure of some agent? Suppose, for instance, the agent is requested to deliver a parcel but the address is wrong. The parcel will not be delivered, but it is not the agent's fault. In such a case, sanction would be pointless because it would not help to achieve the result, and the organization would have no information of the reasons of the failure.

The problem is always the same: *the lack of, broadly speaking, a feedback framework*. Such a lack, for instance, makes it impossible to acquire information about possible conflicts (that remain internal to the agents), and hinders the identification of other agents to which reassign the goal because they have the skills that are needed to cope with a perturbation. As a consequence, the organization will generally be unable of selecting alternative strategies for pursuing its goals in presence of unfavorable conditions. To tackle these conditions effectively as a consequence of a good design, we need new conceptual tools. Inspired by what is often done in human organizations [25,40,46], we claim that *accountability* [10,12,22,27,28] provides such a new tool. As we will discuss in detail in the next section, accountability allows a designer to specify how feedback, that is collected by an agent, is passed to another agent, who is in position to react to the perturbation. Thanks to the former agent's accountability, the other agent, who would not otherwise know of the situation, becomes aware of the perturbation, and triggers its internal deliberative process for deciding how to tackle it (with a straightforward benefit to the whole organization). In our view, accountability is the key to design and develop robust MAS and organizations; we justify this claim in the following section.

3 Robustness Through Accountability

The term accountability has its roots in Latin, where it is related to the verb *computare*, to compute or calculate. Roughly speaking, an accountable person has the capability to provide an account about a condition of interest [21], that is, a person can be accountable for a condition, only if she has some competence, or knowledge, about the very same condition. Accountability "emerges as a primary characteristic of governance where there is a sense of agreement and certainty about the legitimacy of expectations between the community members." [22]. Accountability is, therefore, a mechanism and instrument of administrative and political power. It can be the means through which organizations can ensure the compliance of their processes to predefined standards as well as the force that enables changes aimed at improving the organization [14,25].

In many cultures, accountability is associated with blame [20], either *post factum* (who is to blame for an act or an error that has occurred), or *pre factum* (who is blameworthy for errors not yet occurred), but this is a very partial view that disregards the potential involved in relationships concerning the ability and the designation to provide response about something to someone who is legitimated to ask. In sociology, and in ethnomethodology in particular, it is seen as a basic mechanism that allows individuals to constitute societies [27,38]. Basically, it supports sense-making and coordination in a group of interacting parties, all of whom share an agreement on how things should be done [27], and can be reduced to two key features that connect two parties: one of the parties (the "account taker" or *a-taker*) can legitimately ask, under some agreed conditions, to the other party an account about a process of interest; the other party (the "account giver" or *a-giver*) is legitimately required to provide the account to the a-taker [5,6,19]. We can also say that the relationship between a-giver and a-taker is a relationship between a power-wielder and those holding them accountable, that expresses a general recognition of the *legitimacy of the authority of the parties* that are involved: one to exercise particular powers and the other to hold them to provide an account [28]. Consequently, we see accountability as having two main dimensions:

1. *normative dimension* (expectation), capturing the legitimacy of asking and the availability to provide accounts, yielding expectations on the agents' behavior;
2. *structural dimension* (control), capturing that, for being accountable about a process, an agent must have control over that process and have awareness of the situation it will account for.

Control often is interpreted as the ability to bring about events, possibly through other agents (see e.g., [36,45]), that is, to have power over a situation of interest. In the case of accountability, this means that agents can build the account themselves, either because they were directly involved in the attempt of bringing about some event, or because they can get the information that is necessary to build an account through other agents (see also [12]).

We denote accountability as $\mathbb{A}(x, y, r, u)$, where x is the a-giver, y is the a-taker. When condition r holds, y has the claim-right to ask x for an account about u, and x is in position to provide substantive and authoritative accounts about u. Condition r is not related to the state of u, but rather it represents the circumstance in which the a-giver is held to account (see [12]). When such a condition is not met, the a-giver is not obliged to produce the account. For instance, a buyer may hold a seller to account for some goods, but the seller will have to provide a feedback only if the purchase actually occurred, that is, only if the payment took place. Here, payment is the contextual condition that gives the buyer the right to have the account.

Notably, $\mathbb{A}(x, y, r, u)$ does not imply that x is the agent that brings about u; rather, x must report about the state of u when r holds and a request from y is received. Thus, $\mathbb{A}(x, y, r, u)$ entails an agreement between x and y: x accepts the legitimacy of y to ask about u, as well as, y recognizes the power of x to account about u (normative dimension). Such an account can be produced either because x was involved in first person in the attempt of bringing about u, or because it can reach the information that is necessary to build the account because it plays the role of a-taker in some accountability relationships that concern the parts of u (structural dimension).

3.1 Exemplification in JaCaMo

To accommodate the two dimensions of accountability within a MAO, to the aim of increasing robustness, one needs to operate at different levels of the organization model. First, at the conceptual level, the organization model has to be extended to encompass concepts related to the reporting of facts, and to their treatment. Second, at the normative level, we need to introduce the norms that regulate these new concepts. In particular, robustness relies on delivering feedback about perturbations to agents in charge of handling such perturbations so as to maintain invariant a system property [2]. In the rest of this section, we exemplify a possible realization in the well-known JaCaMo platform [13].

An Organizational Conceptual Model. We exemplify how the accountability dimension can be taken into account within a MAO model by exploiting the conceptual model of JaCaMo [13]. It is worth noting that our approach is not strictly dependent on JaCaMo, but it is applicable in any organizational model where a *Business Task* is structured in terms of organizational goals, or tasks, and where there is an explicit representation of the responsibilities taken up by the agents. To this aim, the conceptual model in Fig. 1 generalizes JaCaMo's concepts `Scheme`, `Mission`, and `Goal` respectively into `Business Task`, `Responsibility`, and `Task` (terms inspired by [26]). The mapping between `Responsibility` and `Mission` deserves some argumentation. In a JaCaMo organization goals are grouped in missions, which are then subject to norms. Specifically, the organization will issue obligations to achieve a mission goal to the agents. The organization can exert such a power on the agents because they are asked to *commit* to a mission at the beginning the execution. That is, if an agent does not fulfill an obligation, the organization is legitimated to sanction the agent by virtue of its commitment to the mission. The rationale is that, since it is not possible, in general, to inspect agents, it is also impossible to know whether the agent possesses the right capabilities to play a role, not even whether the agent will be compliant to the norms. To fill this knowledge gap, agents in JaCaMo are asked to commit to a mission as an implicit declaration that they possess the right behaviors for enacting the mission role, and that they will be receptive to the obligations the organization will issue about the goals in that mission. We interpret such a commitment as a declaration of *responsibility* assumption.

Note that in JaCaMo, an agent fulfills an obligation from the organization by mapping it into an internal goal: the satisfaction of such an internal goal will amount to an achievement of a mission goal, and hence will gain an institutional value. This approach guarantees a strong decoupling between the agents and the organization, allowing the agents to autonomously determine how they accomplish the organizational goals.

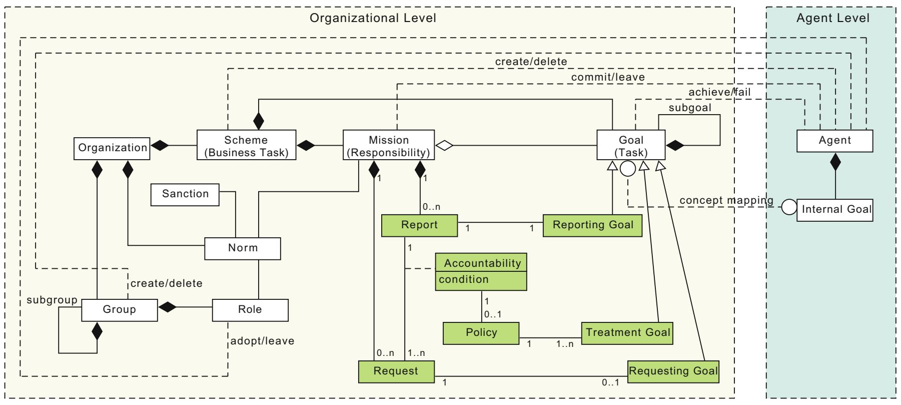

Fig. 1. The enhanced conceptual model. (Color figure online)

Finally, to the sake of generality, in Fig. 1 we also highlight `Sanction` as related to `Norm`, even though in JaCaMo this concept is just implicitly modeled.

Extending the Organization Conceptual Model. The green boxes in Fig. 1 highlights the concepts we add for modeling accountability. We capture the a-giver's side of accountability by means of `Report` as a component of `Mission`. The intuition is that an a-giver provides a report (i.e., an account) which is always contextualized by a mission: a report cannot exist on its own, but it refers to a specific mission to which the a-giver is committed. The association between `Report` and `Reporting Goal` makes it clear that a report is produced by some internal agent goal, mapping the `Reporting Goal`. The result of such an internal goal is a set of facts that gain an institutional meaning as a `Report`.

The a-taker's side of accountability is captured via `Request`, a component of `Mission`. An agent is legitimated to ask for a report only when the mission it is committed to includes at least one `Request`. The right of asking for a report can become an obligation when `Request` is associated with `Requesting Goal`. The organization can, in fact, issue obligations to achieve these goals pushing the agent to act as a-taker.

The relationships between `Report` and `Request` is captured as an association class `Accountability`, whose field `condition` represents the contextual condition that must be satisfied for granting the right of asking a report. It is important to underline that such an association class is usually defined between `Report` and `Request` instances that belong to different missions, and hence are under the responsibility of different agents. In this way, the association models the channel through which a report flows from the a-giver (who produces it) to the a-taker (who uses it). `Accountability` may be related to one (or more) `Policy`, that abstracts a strategy the organization has for copying with a specific report. `Policy`, in turn, is associated with one or more `Treatment Goals` that realize it. These further goals, when defined, are related to the mission of the agent behaving as a-taker: indeed, they capture how the report, provided by the a-giver, is addressed by the a-taker that asked for it.

Accountability Normative and Structural Dimensions. Accountability comes actually into play when the new concepts introduced above are regulated by specific norms. In particular, these norms should not only map the normative dimension of accountability (i.e., the legitimacy –*a-taker*'s side– of asking for an account, and the obligation –*a-giver*'s side– of producing such an account), but also capture its structural dimension. That is, it must be granted that when an agent is obliged to produce a report, that agent has the means for producing an authoritative report, i.e., an *account*.

In JaCaMo, norms are represented and interpreted by the Moise layer by using the Normative Programming Language (NPL) [32]. A norm in this language has the following syntax: `norm` id : φ `->` ψ, where id is an identifier of the norm, φ is the activation condition of the norm, and ψ is the consequence of the norm. A consequence can either be an obligation, or a failure. The former is used to raise obligations toward agents about goals to be achieved. The latter is used to model regimented norms; e.g., conditions that are prohibited. Intuitively, when ϕ is `fail`, any agent action that makes φ true will fail, too (and no change in the organization occurs).

We can reproduce the normative dimension of accountability by means of norms in NPL. For instance, the following norm template induces the accountability $\mathbb{A}(x, y, r, u)$.

```
1 norm reportProduction:
2   accountability(Request_u, Report_u, R) &
3   reportRequest(Y, Request_u) & R &
4   mission(M1, Y) & request(M1, Request_u) &
5   report(M2, Report_u) & mission(M2, X)
6   ->
7   obligation(X, ReportProduction, reportingGoal(Report_u),
8          Deadline)
```

The rule specifies that, when there exists an accountability relating a report about u and a request for the very same report in the context r (line 2), and agent y asks for a report on u under condition r (line 3), and y is legitimated to ask such a report because the request is part of its mission (line 4), and x is competent for producing an authoritative report about u because this is part of its mission (line 5), then an obligation on x is issued about goal `reportingGoal(Report_u)`, through which the agent will provide y with the requested report.

Another norm can be defined to grant y the permission to ask for a report only when the request is part of its mission, and condition r holds. Indeed, in NPL we have to express a norm for prohibiting y to ask for a report when the context does not hold or when it has not a request for that report in its mission.

```
norm requestNotAllowed:
   accountability(Request_u, Report_u, R) &
   reportRequest(Y, Request_u) & ( not R |
        (not (mission(M1, Y) & request(M1, Request_u)))
    ->
     fail (notLegitimateRequest(Y, R, U) )
```

The argument of the `fail` operator, `notLegitimateRequest(Y, R, U)`, represents the reason for the failure.

Following [5], the structural dimension of accountability requires that for each accountability $\mathbb{A}(x, y, r, u)$ defined in the system, either x has control over u, and hence can generate an account by producing facts, or there exists another accountability of the form $\mathbb{A}(z, x, r, u)$ supporting x. In terms of norms, thus, the structural dimension is a property that can be verified by assessing whether for each obligation that agent x has about reporting on u, x has the means for generating a report either from direct control over u, or from a report that x is legitimated (by norms) to ask to another agent. When both the structural and normative dimensions of accountability hold, x is an accountable agent for condition u, that is, x has the power to produce an *account* about u (i.e., an authoritative and reliable collection of facts).

It is worth noting that, although accountability is conceptually a directed relationship between agents, it is realized by means of undirected obligations. This happens because in our discussion we talk about accountability within the context of an organization, and rely on the organization's normative system to realize accountability by way of concepts like obligations and goals. Other approaches, such as [11,18], do not take the organizational perspective, but allow agents to establish their accountability relationships by means of protocols. In these cases, the notion of accountability is usually realized by means of social commitments, that differently from obligations, are always directed from a debtor agent towards a creditor agent.

Adding Robustness Through Accountability. The structural dimension of an accountability $\mathbb{A}(x, y, r, u)$ implies that accountability be grounded on control requirements. However, since it is not generally possible to assume that agents can be inspected, it is also generally impossible to know whether an agent has control over a specific condition when it enacts a role. To fill this knowledge gap, we assume that agents joins an organization only if they take on, explicitly, the *responsibility* of some of the organizational goals. As explained, responsibility is not directly represented in JaCaMo, but we can see the commitment to a mission as a declaration of responsibility assumption. Accountability and responsibility support robustness when the account about a perturbation is reported to the agent who is responsible for treating that perturbation. This is, in fact, a possible mapping of "the feedback interconnection of sensors and actuators" [2] into the organizational setting: the account of a perturbation (feedback) is the response that an a-giver produces as a consequence of a failure of a goal g (perturbation), that is of "interest" to an a-taker. The "interest" stems by the fact that the a-taker is responsible for an organizational goal, G, which cannot be accomplished due to the failure of g. By virtue of its responsibility on G, the a-taker is also responsible for treating any perturbation affecting G. Generally speaking, treating a perturbation means restoring a normal execution flow disrupted by that perturbation, but favorable opportunities could be handled, as well. `TreatmentGoal` abstracts such a task of treating perturbations by means of the mapping with the internal goals of a responsible agent.

4 Related Works

In this paper, we argued that accountability is instrumental for the realization of distributed systems that show some robustness property by design. Other works in literature have advocated the importance of accountability in the design of complex systems. The proposal in [15,16] takes into account Sociotechnical Systems (STS), where multiple, autonomous principals interact with each other. They show how accountability plays a fundamental role in balancing the principals' autonomy. Their point is that accountability does not limit autonomy, since a principal can decide to violate any expectation for which it is accountable. However, by way of accountability, the principal would be held to account about that violation. Accountability relationships have, in fact, a normative stance, and hence they can be used to model the requirements of any STS. Accountability requirements serve as high-level representation of protocols, favoring the modularity of an STS development: a principal just needs to know its accountability requirements, and then can implement its software independently from others.

The work in [17] considers the ethical dimension in the design of STS. The authors argue that social norms provide a standard for correct behavior. Ethics is, in fact, a system-level concern; the point is that whether an agent's actions are ethical depends upon whether the system as a whole is ethical. An ethical system is capable of assessing the violation of an norm, and see it as an opportunity for innovation. An important aspect raised by the authors is that autonomy is not only a matter of intelligence and capabilities, but also involves the ability to violate norms. The rationale is that innovation presupposes the deviation from norms, that is to say, violating norms is not always bad, but sometimes it can lead to improving the whole system. In order to do this, it is necessary to align norms and agents, by relying on explanations that violators are expected to give. So, if the explanation hints a lack in the normative system, the violator is not sanctioned but rather the norms are updated. This approach is pretty different than ours. First of all, there is no explicit distinction between responsibility and accountability. The two concepts are merged within a single notion somehow aligned with liability. In [17], in fact, the normative dimension associated with accountability refers to the expectation that an a-taker has on what the a-giver will do (i.e., be ethical by adhering to norms). When the a-giver does not comply with the expectations, it is implicitly considered responsible for the violation, and hence held to explain the reasons for its behavior.

In our approach, the notion of accountability is not tied to liability, but has a wider understanding, since an accountable agent is not necessarily one to be blamed. To achieve this result we separate the responsibility of action from the accountability about situations [26]. The responsibility to act inside an organization is captured by the commitment to a mission: an agent accepts all the obligations that may be subsequently issued by the normative system of the organization. On the other hand, accountability is characterized by two dimensions: normative and structural. In our case, the normative dimension refers to the expectation an a-taker has on what accounts (i.e., reports) the a-giver is capable to provide. Such a dimension, however, must be supported by the structural dimension, that assures an a-giver has the proper means for producing the accounts it is expected to. Grounding the structural dimension on the assumption of responsibility allows agents to report legitimately about outcomes brought about by other agents [12]. This is essential when, in a distributed system, the perturbation detected by an agent may have to traverse many agents before reaching the one capable of handling it.

Moreover, explanations are not reports: an explanation in [17] is a justification of the agent's norm-violating behavior, while a report, in our understanding, does not have this specific interpretation. Then, a-givers in our approach are not seen as rule-violators: they are agents that meet perturbations and provide information about the encountered situation. The a-takers, on their hand, will interpret the received reports at the callee's level, possibly combining them with further information not available to the agents which met the perturbations. The adaptation process in [17] can, however, be seen as a type of robustness, and hence it bears similarities with the approach presented in the paper. Our objective, however, is not to change the norms, but to support the achievement of the organizational goal despite the occurrence of anticipated perturbations. In [17], instead, accountability enables the process of norms adaptation by feeding outcomes back into the design-phase. The two approaches are not in contrast, rather, they complement each other. They are both exemplifications of the perspective put forward in [2], for which a property of a system is robust if it is invariant with respect to a set of perturbations. The difference lies in the type of perturbations the two approaches aim at.

On the conceptual modeling side, ReMMo (Responsibility MetaModel) by Feltus [26] is one of the few attempts, to the best of our knowledge, of conceptualizing how responsibility can be structured in the frame of an enterprise architecture. There are some interesting similarities, but also substantial differences between ReMMo and the conceptual model we propose. Both in ReMMo and in our proposal, responsibilities originate from (professional) norms agents are held to respect. The two approaches agree that accountability refers to the obligation to report the achievement, maintenance, or avoidance of some given state to an authority [39]. ReMMo relates a responsibility to an aggregate of accountabilities. The rationale is that a responsibility is composed of duties, and an agent assigned to that responsibility is answerable, via accountabilities, for these duties. The same relationship emerges, indirectly, also in our conceptual model. In fact, a `Mission` can be composed by several `Report` and `Request`, and `Accountability` is an association class between them.

There are, however, some important differences. First of all, in ReMMo every responsibility is always associated with one or more accountabilities. We instead allow missions (sources of responsibility) that are not related with reports, and hence with accountabilities. This discrepancy stems from the different aims and scopes of the two models. ReMMo captures the complexity of a human organization, and aims at tracing who is responsible for some task and hence is held to account for what she does (or does not) concerning that task. The goal, thus, is to single out the person(s) who should provide an account for a specific business task performed within the enterprise. In our case, instead, we aim at achieving robustness by way of accountability as a mechanism for modeling feedback flows. But not for every mission feedback may be required, or can possibly be specified, and hence the model lets the definition of missions that are not in relation with reports. In addition, in our model `Report`, `Request`, and `Policy` are explicitly represented not only to specify who has to provide an account to whom, but also how such an account should be used by the a-taker for the robustness purpose. All these concepts are missing in ReMMo, where accountability is substantially overlapped with liability, and hence associated with a sanction. In our view, it is restrictive to see accountability just as a way to find a culprit to be sanctioned; rather, it is an important tool to get a better understanding of what is going on in the system, and possibly take proper action. Sanctions, if necessary, follow from normative decisions, and hence they are associated with the `Norm` concept. This is also the position put forward in [17], where it is observed how sanctions, although may serve as deterrent, remove accountability: by paying its sanction, an agent needs no longer to provide an account about its violation. This of course prevents one to know the causes of the violation, and hence blocks the adaptation process at the basis of robustness.

MOCA [12] is another attempt to model accountability from a computational point of view which deserves some discussion. MOCA is an information model that captures what kind of data (facts) must be available to develop systems that, in any situation of interest arising in a group of interacting agents, permit the identification of account-givers. The model is given in Object-Role Modeling (ORM) [31] due to the relational nature of the represented concepts, and enables automatic verification of consistency of a specific domain description. This allows a designer to establish whether all the relevant pieces of information for supporting accountability have been considered. MOCA builds the information model for accountability around two basic concepts: *just expectation* and *control.* For just expectation it is intended a mutual awareness and acceptance of an accountability relationships between the involved a-giver and a-taker agents. For control, instead, it is intended the power, possibly exerted indirectly by means of other agents, of achieving a condition of interest. These two features are properly captured in our proposal by, respectively, the normative and structural dimensions of accountability. Through the normative dimension, in fact, agents are aware of what obligations they may be subjected as a-giver, and what permissions they have as a-taker. The structural dimension, instead, grounds accountability relationships over an explicit assumption of responsibility from the agents via the commitment to missions. We consider such a commitment as a declaration of direct control (i.e., expressed in MOCA as the relation *can realize*).

Accountability is sometimes put in relation with other properties a system can exhibit, such as transparency, explainability and trust. In particular, the theme of trustful AI is rapidly gaining attention in the last few years. Although some authors consider accountability as opposite to trust [29], others posit that accountability may improve trust when interactions are structured around a clear set of standards [23]. In this paper, we have not focused on this topic, and leave the study of how trust may come into play in our accountability conceptual model to future research. It is worth noting, however, that the two notions are quite different. As we have discussed, accountability is a social relationships between two agents that requires mutual acceptance of rights and duties. Trust, instead, is not necessarily a social relationship: an agent trusts others on the basis of its internal decisions (that usually depend on what others did in the past). For instance, some works propose a strategy for computing trust by assessing how frequently an agent satisfies its commitments [35]. In doing this, however, trust emerges as a local perspective of a single agent, rather than a social relationship.

We conclude this section with a remark about robustness via reactive behaviors. In this paper we have shown how accountability plays a fundamental role in the case of agents because of their autonomy. Autonomy here means that agents are opaque: their beliefs cannot be inspected and their deliberative cycle cannot be known. As pointed out in [17], accountability helps because it defines public relationships that exist outside the agents, regardless of what agents may believe or intend. Of course, there are other settings where robustness can be gained via a purely reactive behavior and where neither accountability nor autonomy come into play. This is, for instance, the case of robustness via control engineering (see e.g., [34]), where a system is modeled as a set of mathematical equations that approximate the system expected outputs for any given inputs. Robustness is gained by means of a controller that, receiving constant feedback of the system outputs, automatically update some system parameters so as to meet its expected performance requirements. This approach is only possible, however, when we are able to design a model of the whole system knowing the (possible approximated) behavior of each of its components. This is not the case in software engineering, however.

5 Conclusions

In this paper, we have posited that an explicit representation of accountability relationships and responsibility declarations form a solid ground upon which a property of robustness can be achieved by design. We have taken into account agent organizations as background of our discussion, and hence we presented an organizational conceptual model apt to capture accountability and responsibility notions. The JaCaMo conceptual model served as a starting point for our extension, but our contribution is not strictly dependent on the JaCaMo model. An important result of this work is a normative characterization for capturing accountability relationships: accountability is, in fact, inherently a normative relationship [17]. We have identified two dimensions featuring accountability:

normative and structural. With the normative dimension we model the relationships between a-takers and a-givers, thus capturing the legitimacy of the former to ask for an account, and the obligation of the latter to provide the account. The structural dimension is instead related to the control, or competence, of a-givers. The structural dimension assures that an agent who is a-giver has the proper means for producing an authoritative account of the situation of interest. Only when such a condition holds, the account represents a meaningful piece of information. We are implementing the approach outlined in the paper by extending the JaCaMo platform, and a first release is on the way.

A key point raised in the paper is that accountability has a positive impact on the agents' autonomy and, due to this, on their adaptability, consequently opening the way to making an organization more robust. On the one side, accountability improves the awareness of the a-taker about what is happening in the system, allowing it to deliberate its (counter)actions accordingly. On the other side, the a-giver's reputation is not automatically reduced when failures occur, because the reports will highlight possible perturbations, supporting the functioning of the organization. Actually, this increases both trust and autonomy [4,42].

This work sets the ground for several future directions. First of all, it represents a general schema that can be tailored to capture specific applications. For instance, it is possible to realize an exception handling mechanism in agent organizations, by constraining the way in which agents produce and consume reports. Specifically, an exception is a situation of interest whose occurrence is related to errors, and which should be urgently reported to the agent in charge of handling these errors. Such "urgency" implies that whenever an agent detects an error it is obliged to report it, even without an explicit request. On the other side, the agent receiving a report will be in charge (i.e., obliged) of tackling the report, that is, handling the exception. These behaviors can be obtained by acting at the normative level of the organization by generating automatically obligations on report and treatment goals. We are currently developing a system, inspired by JaCaMo+ [8] and of [7,9].

In conclusion, we think that the presented framework can be the base for capturing a wide range of non-functional requirements, besides robustness, such as adaptability, fault tolerance, reusability, and transparency. Our intuition is that these non-functional requirements are met in a distributed system when its components (agents in our perspective), can exchange information at a different level of that of the outcomes that are specified by functional requirements. As shown in the paper, accountability can be a valid conceptual tool for reaching this objective.

Acknowledgments. The authors would like to thank the anonymous reviewers for their feedback, that helped to improve the paper, and Olivier Boissier and Stefano Tedeschi for the helpful discussions and support.

References

1. ISO/IEC/IEEE International Standard - Systems and software engineering - Vocabulary. ISO/IEC/IEEE 24765:2010(E), pp. 1–418, December 2010. https://doi.org/10.1109/IEEESTD.2010.5733835
2. Alderson, D.L., Doyle, J.C.: Contrasting views of complexity and their implications for network-centric infrastructures. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **40**(4), 839–852 (2010)
3. Aldewereld, H., Dignum, V., Vasconcelos, W.W.: Group norms for multi-agent organisations. ACM Trans. Auton. Adapt. Syst. **11**(2), 15:1–15:31 (2016)
4. Baarslag, T., Kaisers, M., Gerding, E.H., Jonker, C.M., Gratch, J.: When will negotiation agents be able to represent us? The challenges and opportunities for autonomous negotiators. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 4684–4690 (2017)
5. Baldoni, M., Baroglio, C., Boissier, O., May, K.M., Micalizio, R., Tedeschi, S.: Accountability and responsibility in agent organizations. In: Miller, T., Oren, N., Sakurai, Y., Noda, I., Savarimuthu, B.T.R., Cao Son, T. (eds.) PRIMA 2018. LNCS (LNAI), vol. 11224, pp. 261–278. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03098-8_16
6. Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., Tedeschi, S.: Accountability and responsibility in multiagent organizations for engineering business processes. In: Dennis, L.A., Bordini, R.H., Lespérance, Y. (eds.) EMAS 2019. LNCS (LNAI), vol. 12058, pp. 3–24. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51417-4_1
7. Baldoni, M., Baroglio, C., Capuzzimati, F.: A commitment-based infrastructure for programming socio-technical systems. ACM Trans. Internet Technol. **14**(4), 23:1–23:23 (2014). https://doi.org/10.1145/2677206. http://doi.acm.org/10.1145/2677206
8. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Commitment-based agent interaction in JaCaMo+. Fundamenta Informaticae **159**(1–2), 1–33 (2018)
9. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Type checking for protocol role enactments via commitments. Auton. Agent. Multi-Agent Syst. **32**(3), 349–386 (2018). https://doi.org/10.1007/s10458-018-9382-3
10. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: Computational accountability. In: Chesani, F., Mello, P., Milano, M. (eds.) Deep Understanding and Reasoning: A Challenge for Next-Generation Intelligent Agents, URANIA 2016, Genoa, Italy, vol. 1802, pp. 56–62. CEUR, Workshop Proceedings, December 2016. http://ceur-ws.org/Vol-1802/
11. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: Computational accountability in MAS organizations with ADOPT. Appl. Sci. **8**(4), 489 (2018)
12. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: MOCA: an ORM model for computational accountability. J. Intell. Artif. **13**(1), 5–20 (2019). https://doi.org/10.3233/IA-180014
13. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Sci. Comput. Program. **78**(6), 747–761 (2013). http://www.sciencedirect.com/science/article/pii/S016764231100181X
14. Bovens, M.: Two concepts of accountability: accountability as a virtue and as a mechanism. West Eur. Polit. **33**(5), 946–967 (2010)

15. Chopra, A.K., Singh, M.P.: The thing itself speaks: accountability as a foundation for requirements in sociotechnical systems. In: IEEE 7th International Workshop RELAW. IEEE Computer Society (2014). https://doi.org/10.1109/RELAW.2014.6893477
16. Chopra, A.K., Singh, M.P.: From social machines to social protocols: software engineering foundations for sociotechnical systems. In: Proceedings of the 25th International Conference on WWW (2016)
17. Chopra, A.K., Singh, M.P.: Sociotechnical systems and ethics in the large. In: Furman, J., Marchant, G.E., Price, H., Rossi, F. (eds.) Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2018, New Orleans, LA, USA, 02–03 February 2018, pp. 48–53. ACM (2018)
18. Chopra, A.K., Singh, M.P.: Clouseau: generating communication protocols from commitments. In: Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020), pp. 7244–7252. AAAI Press (2020)
19. Cranefield, S., Oren, N., Vasconcelos, W.W.: Accountability for practical reasoning agents. In: Lujak, M. (ed.) AT 2018. LNCS (LNAI), vol. 11327, pp. 33–48. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17294-7_3
20. Dubnick, M.J.: Blameworthiness, trustworthiness, and the second-personal standpoint: foundations for an ethical theory of accountability. Presented at EGPA Annual Conference, Group VII: Quality and Integrity of Governance, Edinburgh, Scotland, 11–13 September 2013
21. Dubnick, M.J.: Accountability as a Cultural Keyword, pp. 23–38. Oxford University Press, Oxford (2014)
22. Dubnick, M.J., Justice, J.B.: Accounting for accountability, September 2004. https://pdfs.semanticscholar.org/b204/36ed2c186568612f99cb8383711c554e7c70.pdf. Annual Meeting of the American Political Science Association
23. Ehren, M., Paterson, A., Baxter, J.: Accountability and trust: two sides of the same coin? J. Educ. Change **21**(1), 183–213 (2019). https://doi.org/10.1007/s10833-019-09352-4
24. Elder-Vass, D.: The Causal Power of Social Structures: Emergence, Structure and Agency. Cambridge University Press, Cambridge (2011)
25. Executive Board of the United Nations Development Programme and of the United Nations Population Fund: The UNDP accountability system, accountability framework and oversight policy. Technical report DP/2008/16/Rev.1, United Nations (2008)
26. Feltus, C.: Aligning access rights to governance needs with the responsibility metamodel (ReMMo) in the frame of enterprise architecture. Ph.D. thesis, University of Namur, Belgium (2014)
27. Garfinkel, H.: Studies in Ethnomethodology. Prentice-Hall Inc., Englewood Cliffs (1967)
28. Grant, R.W., Keohane, R.O.: Accountability and abuses of power in world politics. Am. Polit. Sci. Rev. **99**(1), 29–43 (2005)
29. Gundlach, G.T., Cannon, J.P.: "Trust but verify"? The performance implications of verification strategies in trusting relationships. J. Acad. Mark. Sci. **38**(4), 399–417 (2010). https://doi.org/10.1007/s11747-009-0180-y
30. Haller, P., Sommers, F.: Actors in Scala - Concurrent Programming for the Multicore Era. Artima, Walnut Creek (2011)
31. Halpin, T., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann Publishers, Burlington (2008)

32. Hübner, J.F., Boissier, O., Bordini, R.H.: A normative programming language for multi-agent organisations. Ann. Math. Artif. Intell. **62**(1), 27–53 (2011). https://doi.org/10.1007/s10472-011-9251-0
33. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Auton. Agent. Multi-Agent Syst. **13**(2), 119–154 (2006)
34. Ioannou, P.A., Sun, J.: Robust Adaptive Control. Courier Corporation, North Chelmsford (2012)
35. Kalia, A.K., Zhang, Z., Singh, M.P.: Estimating trust from agents' interactions via commitments. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) ECAI 2014 - 21st European Conference on Artificial Intelligence, 18–22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014). Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 1043–1044. IOS Press (2014). https://doi.org/10.3233/978-1-61499-419-0-1043
36. Marengo, E., Baldoni, M., Baroglio, C., Chopra, A., Patti, V., Singh, M.: Commitments with regulations: reasoning about safety and control in REGULA. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), vol. 2, pp. 467–474 (2011)
37. Platon, E., Sabouret, N., Honiden, S.: Challenges for exception handling in multi-agent systems. In: Choren, R., Garcia, A., Giese, H., Leung, H., Lucena, C., Romanovsky, A. (eds.) SELMAS 2006. LNCS, vol. 4408, pp. 41–56. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73131-3_3
38. Rawls, A.W.: Harold Garfinkel, ethnomethodology and workplace studies. Organ. Stud. **29**(5), 701–732 (2008)
39. Sommerville, I., Lock, R., Storer, T., Dobson, J.: Deriving information requirements from responsibility models. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 515–529. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02144-2_40
40. Sustainable Energy for All Initiative: Accountability framework. https://sustainabledevelopment.un.org/content/documents/1644se4all.pdf
41. Timm, I.J., Scholz, T., Herzog, O., Krempels, K.H., Spaniol, O.: From agents to multiagent systems. In: Kirn, S., Herzog, O., Lockemann, P., Spaniol, O. (eds.) Multiagent Engineering. INFOSYS, pp. 35–51. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-32062-8_3
42. Winikoff, M.: Towards trusting autonomous systems. In: El Fallah-Seghrouchni, A., Ricci, A., Son, T.C. (eds.) EMAS 2017. LNCS (LNAI), vol. 10738, pp. 3–20. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91899-0_1
43. Woods, D.D.: The risks of autonomy: Doyle's catch. J. Cogn. Eng. Decis. Mak. **10**(2), 131–133 (2016)
44. Wooldridge, M., Jennings, N.R., Kinny, D.: The GAIA methodology for agent-oriented analysis and design. Auton. Agent. Multi-Agent Syst. **3**(3), 285–312 (2000). https://doi.org/10.1023/A:101007191086910.1023/A:1010071910869
45. Yazdanpanah, V., Dastani, M.: Distant group responsibility in multi-agent systems. In: Baldoni, M., Chopra, A.K., Son, T.C., Hirayama, K., Torroni, P. (eds.) PRIMA 2016. LNCS (LNAI), vol. 9862, pp. 261–278. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44832-9_16
46. Zahran, M.: Accountability Frameworks in the United Nations System (2011). https://www.unjiu.org/sites/www.unjiu.org/files/jiu_document_files/products/en/reports-notes/JIU%20Products/JIU_REP_2011_5_English.pdf. UN Report
47. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: the Gaia methodology. ACM Trans. Softw. Eng. Methodol. **12**(3), 317–370 (2003)