Fragility and Robustness in Multiagent Systems

Matteo Baldoni[1 [0000−0002−9294−0408]], Cristina Baroglio[2,3 [0000−0002−2070−0616]], and Roberto Micalizio[3 [0000−0001−9336−0651]]

Università degli Studi di Torino, Dipartimento di Informatica
firstname.lastname@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 for building 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 [26]. 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 [1]. 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 [1], 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., [32, 29, 2]) 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 (see [18]). This happens because traditional approaches to exception handling do not accommodate some important features of MAS, like openness, heterogeneity, agent encapsulation, and distribution [22] – e.g., the common assumption is that software components are "collaborative", and that the code will be available for inspection. Following [22], in the case of MAS similar mechanisms should leverage on the proactivity of agents.

We can, however, see in action elements that support the systematic introduction of robustness. For instance, often agents rely on reputation and trust to estimate how reliable another agent is before using a piece of information, that was produced by that agent, in their deliberative cycle [20]. 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 needed. Alderson and Doyle [1] 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 Section 3 we introduce accountability and lay the basics for building reporting frameworks through it. Finally, in Section 3.1 we introduce a possible architecture, extending that of JaCaMo [7], 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" [28]. In sociology, such a complex network becomes a set of constraints that make a system, which comprises many parts, to act as a whole [13]. 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). 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" [1]. For example, consider the autonomous vehicle

described in [28]. 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 taking advantage of the contribution of each agent [25]. An organization is, thus, 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 an agent to achieve that goal. Agents' acceptance of the organizational constraints enables the agents themselves to act in a shared environment, and achieve results unachievable if they acted in isolation. The agents' autonomy is an enabler of the system's 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 towards the misbehaving agent. The normative system, thus, is both the means that enables the orchestration of the activities of a group of autnomous agents, and in some sense it is also the means that tries to produce robustness, in that agents are pushed to do what is expected of them, and thus to tackle the situations the system faces. The rationale is to guide the agents toward the interest of the organization.

Generally, however, sanctions are not 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 agents' self-interest in pursuing different goals of their own, and (2) target agents that actually have the resources and the capabilities, that are 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 design, conceptual tools. We claim that the concept of *accountability* [15, 17, 12, 5, 6] is such a new tool, similarly to what is often done in human organizations [24, 31]. 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 deep 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 [11], 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." [12]. 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 improve the organization [8].

In many cultures, accountability is associated to blame [10], 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 [15, 23]. Basically, it supports the sense-making and coordination in a group of interacting parties, all of whom share an agreement on how things should be done [15], 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 [4, 9]. 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 [17]. 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., [21, 30]), 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 [6]).

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. Notably, $\mathbb{A}(x, y, r, u)$ does not imply that x actually 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 recogizes 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 such an account because it plays the role of a-taker in some accountability relationships that concern the parts of u (structural dimension).

We believe accountability to be a kind of constraint that "deconstrains"[1], that is, which helps to build robust MAO by leveraging on the adaptability and autonomy of the involved parties. On the agent's side, the "cost" of accountability is represented by the acceptance of the agreement: one party has the legitimate right to ask for an account, and the other party is held to provide the account, provided that the contextual condition holds. On the organization's side, the cost is to devise those norms that build the structural dimension of accountability.

[1] Gerhart and Kirchner, in explaining biological systems [16], maintain that certain kinds of constraints, indeed, *deconstrain* the interacting components, because they provide adequate support for adaptability. In other words, "constraints that deconstrain" are frameworks that support robustness and evolvability (of which adaptability is a special case) to the price of accepting certain fixed constraints.

Accountability generally has a positive impact on the autonomy of the involved agents and, due to this, on their adaptability, thus opens the way to making the system, made of interacting agents, more robust. The a-taker, indeed, based on the reports provided by the a-giver, can better understand what is happening in the system, and deliberate how to act accordingly. The a-giver's reputation, on the other side, is not automatically reduced when failures occur, because the reports will highlight the real situation, supporting the functioning of the organization. Actually, this increases both trust and autonomy [27, 3].

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 [1]. In the rest of this section, we exemplify a possible realization in the well-known JaCaMo platform [7].

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 [7]. 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 Figure 1 generalizes JaCaMo's concepts `Scheme`, `Mission`, and `Goal` respectively into `Business Task`, `Responsibility`, and `Task` (terms inspired by [14]). 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. This is a simplification because taking on the responsibility of a goal has much stronger implications, but it is acceptable to the aims of an exemplification.

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.

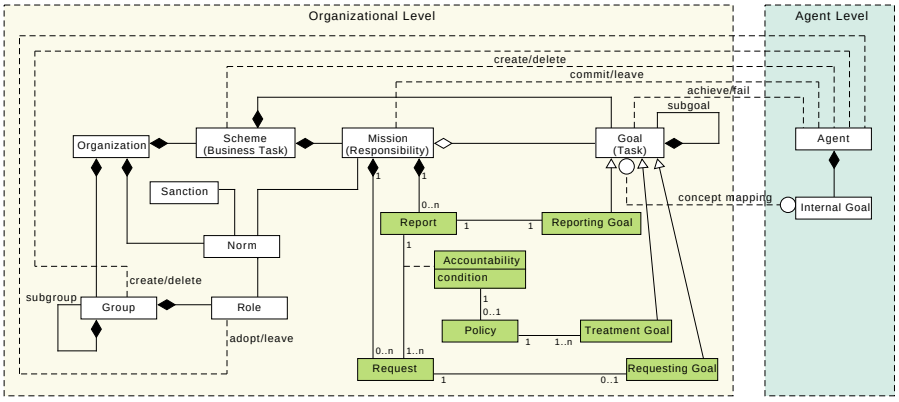

Fig. 1. The enhanced conceptual model.

Finally, to the sake of generality, in Figure 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 Figure 1 highlights the concepts we add for the modeling the accountability dimension. 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 to which it is committed 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 Goal`s 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 map not only 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 its structural dimension. That is, it must be granted that when an agent receives an obligation of producing 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) [19]. 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, given the accountability $\mathbb{A}(x, y, r, u)$, the following norm template.

```
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 towards 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.

```
1  norm requestNotAllowed:
2    accountability(Request_u, Report_u, r) &
3    reportRequest(y, Request_u) & ( not r |
4      (not (mission(m1, y) & request(m1, Request_u)))
5  ->
6  fail (notLegitimateRequest(y, r, u) )
```

The argument of the `fail` operator, `notLegitimateRequest(y, r, u)`, represents the reason for the failure.

Following [4], 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 receive 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).

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" [1] 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 of 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. This task is abstracted by the `TreatmentGoal` that we have introduced at the conceptual level and that may become an internal goal of a responsible agent.

4 Conclusions

We have outlined how accountability can be a design tool for achieving robustness – by properly defining norms, it is possible to issue automatic obligations on report and treatment goals. This may also be the key to implement exception handling within the agent paradigm. Finally, the presented framework can be the base for capturing a wide range of non-functional requirements, besides robustness, such as adaptability and transparency.

Acknowledgements

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

References

1. Alderson, D.L., Doyle, J.C.: Contrasting views of complexity and their implications for network-centric infrastructures. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans **40**(4) (2010)
2. Aldewereld, H., Dignum, V., Vasconcelos, W.W.: Group Norms for Multi-Agent Organisations. ACM Trans. Auton. Adapt. Syst. **11**(2), 15:1–15:31 (Jun 2016)
3. 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, August 19-25, 2017. pp. 4684–4690 (2017)
4. Baldoni, M., Baroglio, C., Boissier, O., May, K.M., Micalizio, R., Tedeschi, S.: Accountability and Responsibility in Agents Organizations. In: PRIMA 2018: Principles and Practice of Multi-Agent Systems, 21st International Conference. pp. 403–419. No. 11224 in Lecture Notes in Computer Science, Springer, Tokyo, Japan (October 31st–November 2nd 2018)
5. 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. vol. 1802, pp. 56–62. CEUR, Workshop Proceedings, Genoa, Italy (December 2016), http://ceur-ws.org/Vol-1802/
6. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: MOCA: An ORM MOdel for Computational Accountability. Journal of Intelligenza Artificiale **13**(1), 5–20 (2019). https://doi.org/http://dx.doi.org/10.3233/IA-180014

7. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming **78**(6), 747 – 761 (2013), http://www.sciencedirect.com/science/article/pii/S016764231100181X
8. Bovens, M.: Two concepts of accountability: Accountability as a virtue and as a mechanism. West European Politics **33**(5), 946–967 (2010)
9. Cranefield, S., Oren, N., Vasconcelos, W.: Accountability for practical reasoning agents. In: AT 2018: 6th International Conference on Agreement Technologies. LNCS (2018)
10. 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)
11. Dubnick, M.J.: Accountability as a Cultural Keyword, pp. 23–38. Oxford University Press (2014)
12. 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
13. Elder-Vass, D.: The Causal Power of Social Structures: Emergence, Structure and Agency. Cambridge University Press (2011)
14. Feltus, C.: Aligning Access Rights to Governance Needs with the Responsability MetaModel (ReMMo) in the Frame of Enterprise Architecture. Ph.D. thesis, University of Namur, Belgium (2014)
15. Garfinkel, H.: Studies in ethnomethodology. Prentice-Hall Inc., Englewood Cliffs, New Jersey (1967)
16. Gerhart, J., Kirschner, M.: The theory of facilitated variation. Proceedings of the National Academy of Sciences of the United States of America (PNAS) **104 (suppl 1)** (2007)
17. Grant, R.W., Keohane, R.O.: Accountability and Abuses of Power in World Politics. The American Political Science Review **99**(1) (2005)
18. Haller, P., Sommers, F.: Actors in Scala - concurrent programming for the multicore era. Artima (2011)
19. Hübner, J.F., Boissier, O., Bordini, R.H.: A normative programming language for multi-agent organisations. An. of Math. and Art. Intel. **62**(1), 27–53 (2011)
20. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Autonomous Agents and Multi-Agent Systems **13**(2), 119–154 (2006)
21. Marengo, E., Baldoni, M., Baroglio, C., Chopra, A., Patti, V., Singh, M.: Commitments with regulations: reasoning about safety and control in REGULA. In: Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). vol. 2, pp. 467–474 (2011)
22. Platon, E., Sabouret, N., Honiden, S.: Challenges for exception handling in multiagent systems. In: Software Engineering for Multi-Agent Systems V. pp. 41–56. Springer, Berlin, Heidelberg (2007)
23. Rawls, A.W.: Harold Garfinkel, Ethnomethodology and Workplace Studies. Organization Studies **29**(701) (2008)
24. Sustainable Energy for All Initiative: Accountability framework, https://sustainabledevelopment.un.org/content/documents/1644se4all.pdf
25. Timm, I.J., Scholz, T., Herzog, O., Krempels, K., Spaniol, O.: From agents to multiagent systems. In: Kirn, S., Herzog, O., Lockemann, P.C., Spaniol, O. (eds.) Multiagent Engineering, Theory and Applications in Enterprises, pp. 35–51. Springer (2006). https://doi.org/10.1007/3-540-32062-8_3

26. VV.AA.: Iso/iec/ieee international standard - systems and software engineering – vocabulary. ISO/IEC/IEEE 24765:2010(E) pp. 1–418 (Dec 2010). https://doi.org/10.1109/IEEESTD.2010.5733835
27. Winikoff, M.: Towards Trusting Autonomous Systems. In: Engineering Multi-Agent Systems - 5th International Workshop, EMAS 2017, Sao Paulo, Brazil, May 8-9, 2017, Revised Selected Papers. pp. 3–20 (2017)
28. Woods, D.D.: The risks of autonomy: Doyle's catch. Journal of Cognitive Engineering and Decision Making **10**(2) (2016)
29. Wooldridge, M., Jennings, N.R., Kinny, D.: The GAIA methodology for agent-oriented analysis and design. Autonomous Agents and multi-agent systems **3**(3), 285–312 (2000)
30. Yazdanpanah, V., Dastani, M.: Distant group responsibility in multi-agent systems. In: PRIMA 2016: Principles and Practice of Multi-Agent Systems - 19th International Conference, Phuket, Thailand, August 22-26, 2016, Proceedings. pp. 261–278 (2016). https://doi.org/10.1007/978-3-319-44832-9_16
31. Zahran, M.: Accountability Frameworks in the United Nations System. https://www.unjiu.org/sites/www.unjiu.org/files/jiu_document_files/products/en/reports-notes/JIU%20Products/JIU_REP_2011_5_English.pdf (2011), uN Report
32. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Trans. Softw. Eng. Methodol. **12**(3), 317–370 (2003)