Accountability, Responsibility and Robustness in Agent Organizations

Matteo Baldoni[0000-0002-9294-0408] (✉), Cristina Baroglio[0000-0002-2070-0616], and Roberto Micalizio[0000-0001-9336-0651]

Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino (Italy)
firstname.lastname@unito.it

Abstract. Agent organizations are widely used for the design and development of distributed systems. Agent organizations, however, lack of a systematic way to treat exceptions. In this paper we introduce ARFIN organizations as a novel way to conceptualize multiagent systems. We characterize ARFIN organizations in terms of *accountability* and *responsibility* notions that complement the normative system of a standard organization. ARFIN improves the robustness of the resulting system since it enables a systematic treatment of exceptions by detecting and reporting them to the agent with the right capabilities for handling them.

Keywords: Accountability · Responsibility · Agent Organizations · Engineering MAS.

1 Introduction

Agent organizations [16, 19, 11] are a well-known abstraction for modularizing the software by functionally decomposing the organizational goal into subgoals. These are, then, assigned to the agents by the organization through obligations. Obedience, however, cannot be given for granted. Agents will satisfy an obligation only when this is functional to the achievement of their own goals, and faults may happen. This is not itself an issue, the issue is that organizations provide no means for systematically tackling the information that made execution deviate from what expected. An interesting perspective is drawn by works, like [8, 6, 3], that argue that agent-based software should be modularized in terms of the accountabilities that each agent autonomously takes on, concerning the goals it is expected to pursue.

The problem has implications on software engineering in general and becomes more relevant due to the fact that we witnessing an increasing spreading of "intelligent and autonomous" software in many aspects of our everyday life. Indeed, we already entrust software, sometimes even without noticing, with critical decisions in several scenarios. Modern planes, for instance, are equipped with (software) autopilots that not only help pilots in keeping the route, but even override the human pilots when their decisions might endanger the safety of the aircraft. It is therefore evident that, when intelligent software has the power to take decisions autonomously, we have a serious problem of making such software *accountable*, that is, under given conditions it provides accounts of what was achieved or what went wrong.

Trying to make an existing software accountable might be an arduous task. It is convenient to think of accountability since the design throughout the development phases of software. In this paper we present some preliminary results (see [3, 4]), and some interesting perspectives about a novel way to conceptualize accountable software. Specifically, we take into account software that is distributed in nature and that can be seen as a multiagent system (MAS) organization (e.g., [1]): independent components (i.e., agents) acting concurrently and that cooperate for achieving a global goal. Agent organizations represent strategies of decomposing complex organizational goals into simpler sub-tasks and allocating them to roles. By adopting roles in the organization, agents take on responsibilities and execute the corresponding tasks in a distributed, coordinated and regulated fashion.

Current models, although targeting open systems by allocating and enforcing rights and duties to agents about the tasks to realize, lack an explicit representation of directed relationships between the agents that allows the agents to identify who should give restitution to whom for a certain state of the organization. Such a representation would overcome a current lack of agent organizations, that is, even if agents who enter the organization are under the regulation of norms, that stipulate their rights and duties, the organization has no means for asking agents feedback about their conduct. Feedback that, reported to those who have the claim right of obtaining it, can either be used for certification purposes, or for tackling exceptional situations.

We claim that the realization of accountable software should be grounded on the explicit representation of two fundamental concepts at the basis of human organizations: accountability and responsibility.

2 ARFIN Organizations

We think of a MAS organization as a process being collectively executed by a number of involved agents, who generally depend on one another for carrying out their activities. In their interaction agents produce and answer to institutional events, and need to coordinate to accomplish the organizational goal. We now introduce the key concepts at the core of an ARFIN MAS, which are *accountability*, *responsibility*, *fitting*, and *norms*.

According to Grant and Keohane [13], accountability implies that some actors have the right to hold other actors to a set of standards, to judge whether they have fulfilled their responsibilities in light of these standards, and to impose sanctions if they determine that these responsibilities have not been met. They explain that accountability presupposes a relationship between power-wielders and those holding them accountable, where there is a general recognition of the legitimacy of (1) the operative standards for accountability and (2) the authority of the parties to the relationship (one to exercise particular powers and the other to hold them to account). Still in [13], accountability mechanisms always operate after the fact (ex post), by exposing actions to view, judging and sanctioning them. However, due to anticipation of sanctions, accountability mechanisms can exert effects also ex ante, directing behavior towards the standard [2]. In the light of this understanding, and building upon [3], we explicitly represent accountabilities as $\mathsf{A}(x, y, r, u)$, meaning that x, the account-giver, is accountable towards y, the account-taker, for the condition u when the condition r (*context*) holds. Both r

and u are temporal expressions, given in precedence logic [20]. Each accountability yields both a *permission* and an *obligation*: the permission to y to ask for an account of u when the condition r is true, and the obligation to x to provide such an account when y has the claim right to ask for it. Through accountability x states its awareness of such a permission and obligation, and this is enough for y to expect and call x to behave up to such standard. This in turn produces a directed relationship between x and y, where x is liable towards y of providing an account about u when the condition r holds. Note that accountability does not create an obligation on x to accomplish u.

The second key component is *responsibility*, a polyhedric term; in particular, [22] proposes an ontology relating six different responsibility concepts (capacity, causal, role, outcome, virtue, and liability), stemming from the parable of "Smith the captain" by the legal philosopher Herbert L.A. Hart. Roughly speaking, they respectively amount to: doing the right thing, having duties, an outcome being ascribable to someone, a condition that produced something, the capacity to understand and decide what to do, something being legally attributable. We see responsibility assumption as a declaration, by an agent, of being a recipient for (and being moved by) some institutional event. Nevertheless, responsibility does not imply that the agent is expected to provide any feedback, even when (as it is often the case) the agent is the one who is intended to answer to the institutional event by carrying out some activity, by enacting some behavior, or by trying to pursue a given goal.

Responsibility does not entail either that the agent has the capabilities for carrying out a given task. Still, when the agent does have them, it may not always be willing to act as expected, or it may fail in the attempt.

Institutional events are generated by a *normative system*, that is capable of generating obligations, permissions, and so forth, depending on the occurrences of events in the physical and in the institutional world. Obligations, however, are not enough, see [8, 6], because agents not necessarily will accept them. Whenever the agent's goals are in conflict with the organization's goals, and the expected sanction is acceptable, the agent will ignore the obligation. Through responsibility assumptions (that are provided by the agents themselves) we identify those agents who are receptive of the obligation, but it is only by way of accountability (that is also provided by the agents) that it becomes possible to hold agents to account for their execution. The link between responsibilities and accountabilities is realized through the *accountability fitting* (*fitting* for short, [3]). So, on the issuing of some obligation, not only will an agent be receptive, but under given conditions, it will also be liable to a second agent who, by way of the permissions granted by accountability, has the authority for asking the first agent an account. Through the account it will be possible to identify lacks in the system – for instance, an agent not having the capabilities for handling a case, or harm due to exposure to unmanaged exogenous conditions.

For brevity, we refer to a MAS organization which includes all such elements as an *ARFIN organization*. ARFIN organizations are particularly interesting for two main reasons. First, because they meet the requirements that are posed on agent coordination by agent autonomy. In absence of introspection, agent autonomy, in fact, calls for a way of conceptualizing coordination where agents are clearly constrained in terms of responsibilities (that they explicitly take on), and where agents establish directed relationships

from one to another, that reflect the legitimate expectations the second principal has of the first. Second, because they support system robustness. In [12], robustness in software systems is defined as "the ability of a software to keep an 'acceptable' behavior, expressed in terms of *robustness requirements*, in spite of *exceptional* or *unforeseen* execution conditions (such as the unavailability of system resources, communication failures, invalid or stressful inputs, etc.)." Casting such a definition in the context of organizations, an organization is *robust* when its agents can properly react to unexpected events. The driver of such a process is the attempt to execute up to the preset standards, possibly through self-regulation, by adapting either the execution or the organization itself to initially unforseen cases. This process heavily relies on the accounts that the involved agents are expected to produce.

3 Accountability and Responsibility for Exception Handling

Agent organizations are generally specified in terms of *roles*, which have to be adopted by agents, *tasks* (e.g., actions, goals, interactions) assigned to roles through *norms* [9, 7]. Tasks are organized according to a functional decomposition that specifies how a complex, organizational goal can be achieved by work distribution and coordination. Current organizational models do not capture directly the notion of accountability and responsibility as described above. In our opinion, this hampers the development of robust organization.

Specifically, our claim is that the fundamental role of accountability in software development (together with responsibility) is the realization of robust systems. Following [18], on software design, a useful question is "What events are coming in to our system? Why? Because we have to design the software to handle these events [...] Basically, a software system reacts to three things: 1) external events from actors (humans or computers), 2) timer events, and 3) faults or exceptions". Robustness is achieved by assuring, by design, that exceptional events, when occurring, are reported to the agents who can handle them properly. Accountability fulfills this purpose because, by nature, it brings about an obligation on the a-giver to give an account of what it does. The account, then, can be used by the a-taker to recover, when possible, from the exceptional situation. Potentially, the a-taker could pass the account up to other agents for whom it is an a-giver. We, thus, propose to complement the functional decomposition of the organizational goal, that is the backbone of the underlying normative system, with a set of accountability and responsibility specifications.

Considering an accountability $\mathsf{A}(x, y, r, u)$, if we think of a process being collectively executed, we can say that when the r part of the process is done, then x becomes accountable of the u part. When u is true, x is considered to have built a proof that can be supplied to the account-taker. A proof, here, is what intended as a set of recorded facts, that demonstrate the achievement of the specified condition. When, instead, u is false, the agent is once again expected to provide a proof on request of what happened. When r is false, instead, the condition under which the a-taker has the claim right to ask for a proof do not hold anymore, and the a-giver will not have the obligation to provide a proof on request. Instead, intuitively with $\mathsf{R}(x, q)$, x declares to accept to be considered in the position for causing q. This may imply that x has the capacity of producing

q directly, or that x can exert some control on some other agent that will bring about q. We denote by **A** a set of accountabilities, calling it an *accountability specification*, and by **R** a *responsibility distribution*, that is a set of responsibility assumptions that complement the specification of an agent organization.

In general, the fact that a group of agents is compliant to a given fitting provides robustness to the system of interacting agents. From an implementation perspective, it is convenient to focus on the set of possible events that are relevant to the purpose of provinding an account. We say that the a-taker is robust with respect to certain events when, on their occurrence, it proactively provides an account to the a-giver, without waiting to be asked for. Just to give a hint of the technical setting, we use residuation [5] to compute the progress of both accountabilities and responsibility assumptions. Let $\mathcal{U}$ be the universe of discourse (the set of events upon which temporal expressions are formed). Let $\boldsymbol{e}$ be a sequence of events in $\mathcal{U}$, then $\mathsf{A}(x, y, r/\boldsymbol{e}, u/\boldsymbol{e})$ denotes the residual of $\mathsf{A}(x, y, r, u)$ with respect to $\boldsymbol{e}$ (intuitively what remains after the occurrence of the sequence of events $\boldsymbol{e}$). Residuation is the mechanism through which the permission and obligation that are naturally involved in an accountability are triggered. When $r/\boldsymbol{e} \doteq 0$, the accountability is no longer meaningful; when $r/\boldsymbol{e} \doteq \top$, y has the permission to ask x an account about u, and x, in case of a request, is obliged to provide such an account, in particular when the condition is achieved ($u/\boldsymbol{e} \doteq \top$) and when something goes wrong ($u/\boldsymbol{e} \doteq 0$). In the first case, the account is u itself. In the second case, instead, sequence $\boldsymbol{e}$ contains at least one event e which is complementary to some other event in u: the occurrence of e prevents the satisfaction of u. Upon request from y, x will be obliged to provide an account about the unfulfillment of u, and hence it will be obliged to disclose the occurrence of the exceptional event e.

Let us call *actualization* of a temporal expression q any sequence $\boldsymbol{e}$ of events in $\mathcal{U}$ such that $q/\boldsymbol{e} \doteq \top$; we denote such an actualization as $\widehat{q}$.

Definition 1 (Accountability fitting [3]). *Given an accountability specification* **A** *and a responsibility distribution* **R***, we say that* **R** fits **A***, denoted by* $\mathbf{R} \rightsquigarrow \mathbf{A}$*, if for each accountability* $\mathsf{A}(x, y, r, u) \in \mathbf{A}$*, there is a responsibility* $\mathsf{R}(x, q) \in \mathbf{R}$ *such that* $(u/r)/\widehat{q} \equiv \top$*, for some actualization* $\widehat{q}$*.*

Intuitively, the fitting $\mathbf{R} \rightsquigarrow \mathbf{A}$ relates a responsibility declaration $\mathsf{R}(x, q) \in \mathbf{R}$ to one (or even more) accountability relationship $\mathsf{A}(x, y, r, u)$ that is supported by that responsibility via one of its actualizations. The fitting has an important consequence on the design of the organization: when condition $\mathbf{R} \rightsquigarrow \mathbf{A}$ holds, there exists at least one path, along the functional decomposition of the goal, that is covered by accountabilities and responsibility assumptions [3]. This means that, along that path, there is always a role that takes on a responsibility for a task, and accounts for it. Fitting guarantees that an organization is specified coherently. In addition, it is also a specification of a robust organization in the sense that exceptional events are reported.

We, then, propose to characterize fitting with the *exceptional events* that, for domain-dependent reasons, are deemed critical. Let $\mathcal{E}$ be a set of exceptional events, that is, $\mathcal{E} \cap \mathcal{U} = \emptyset$ and each event $e \in \mathcal{E}$ is complementary to some event in $\mathcal{U}$. In general, the same event e could be considered complementary to many events in $\mathcal{U}$. Relation $\mathcal{F} \subseteq \mathcal{U} \times \mathcal{E}$ maps events in $\mathcal{U}$ to their corresponding complementary ones in $\mathcal{E}$. We

say that an expression u is *touched* by an exception $e \in \mathcal{E}$ if for at least one event w occurring in u, $(w, e) \in \mathcal{F}$. By extension, an accountability relationships $\mathsf{A}(x, y, r, u)$ is touched by the occurrence of event e when w occurs in u and $(w, e) \in \mathcal{F}$. Let $[\mathbf{R} \rightsquigarrow \mathbf{A}]_{\mathcal{F}}$ be an accountability fitting characterized by $\mathcal{F}$. An ARFIN organization is compliant with $[\mathbf{R} \rightsquigarrow \mathbf{A}]_{\mathcal{F}}$ if, whenever $\mathsf{A}(x, y, r, u) \in \mathbf{A}$ is touched by an event $e \in \mathcal{E}$, an account about u is requested to x by default.

4 Discussion

Normative multiagent systems (NorMAS) [17, 7] have been widely studied in the research area on MAS. According to [7] a NorMAS is: "a multiagent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms". Interestingly, norms specify either institutional actions, or the conditions for the use of such actions, consequently regulating the acceptable behavior of the agents in a system ("doing the right thing" rather than "doing what leads to a goal" [21]). Agent organizations, such as those modeled in $\mathcal{M}$OISE [15], are a special case of NorMAS which basically rely on obligations for the progression of the execution. Intuitively, the normative system is in charge of issuing obligations to agents at the right times. If the agents accomplish their tasks, the execution progresses toward the organizational goal. When agents fail to satisfy an obligation, there is no mechanism for handling the failure. In most cases, a violation leads to sanctioning the liable agent, but sanctions are deterrents, that should prevent an agent from acting outside the norms, and do not support any recovery mechanism. Practically adopted solutions are not always effective. In $\mathcal{M}$OISE, for instance, the organization keeps issuing to the same agent the unsatisfied obligation. If the causes of the failure are outside the scope of the agent, however, re-emitting the obligation will not solve the problem.

Commitment-based protocols, e.g. [23], provide an alternative for modeling coordination. Roughly speaking, a commitment is a promise that a debtor does in favor to a creditor that in case some antecedent condition is satisfied, the debtor will bring about a consequent condition. When the antecedent holds, the commitment is detached, and amounts to an obligation on the debtor to bring about the consequent. When the consequent is no longer achievable, the commitment is violated. In such a case, the creditor has the right to complain against the debtor, however, the creditor cannot hold the debtor to provide an explanation. This lack of information hampers both the understanding of what has actually occurred, and any attempt of recovery from the failure. The introduction of accountability to this picture would help to overcome such limits, due to the obligation, on the a-giver, to provide feedback about the consequent under certain conditions.

Both agent organizations and commitment-based protocols, thus, tend to be fragile since exceptions cannot be handled systematically. The accountability fitting we have discussed here, instead, supports robustness by allowing a systematic management of exceptions. Notably, the fitting does not substitute the normative system, but rather the fitting flanks the organization norms and the functional decomposition of its goal, as-

suring that, when necessary, it is possible to get information (e.g., explanations) out of agents. Indeed, accountabilities create obligations to provide accounts for conditions of interest, provided that some contextual conditions hold. The account is in first lieu captured by the a-taker, who can try to recover from the failure or, as a second possibility, forward the account to its a-taker (if any). Responsibilities play their part in the picture: when an agent enacts a role and assumes all the responsibilities that come along, the agent declares to be aware of the duties it may be asked to accomplish within the organization. In software terms, the agent declares to have a plan that is triggered by an institutional event (e.g., the issuing of an obligation), and that will bring about any task falling under its responsibilities. Of course, this is just a declaration from the agent, not a verification of its correct behavior.

Also [10] recognizes the value of accountability in the development of software and makes a proposal that is complementary to ours. Specifically, the authors focus on the issue of answer production in presence of an accountability relationship, a problem that involves: how to properly define the temporal window to consider? Which pieces of information are relevant and, thus, are to be kept in this temporal interval? Which questions are suitable to be asked in this setting? The account giving agent produces an answer in terms of its internal mechanisms. What that proposal does not provide is the organizational view of the system of interacting agents and they do not tackle robustness and exceptions.

ARFIN organizations set the ground for developing a mechanism for handling exceptions in agent-based systems. A reference strategy is the one exploited in the actor model (e.g., [14]): an actor usually creates other child-actors assigning them specific tasks. When a child-actor cannot handle a situation and gets an exception, it usually reports the exception to its parent actor. The rationale is that the parent actor is the one interested of having the task done, and possesses the right information to handle the exception properly. In an agent-based system such a mechanism is not directly applicable for agents are independent entities, and are not related by a parent-child relationship. Accountabilities can fill in this gap.

Acknowledgements

The authors would like to thank Olivier Boissier for the stimulating discussions.

References

1. Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., Padget, J. (eds.): Social Coordination Frameworks for Social Technical Systems, Law, Governance and Technology Series, vol. 30. Springer International Publishing (Aug 2016). https://doi.org/10.1007/978-3-319-33570-4, https://hal-emse.ccsd.cnrs.fr/emse-01355372
2. Anderson, P.A.: Justifications and precedents as constraints in foreign policy decision- making. American Journal of Political Science **25**(4) (1981)
3. Baldoni, M., Baroglio, C., Boissier, O., May, K.M., Micalizio, R., Tedeschi, S.: accountability and responsibility in agent organizations

4. Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., Tedeschi, S.: Accountability and Agents for Engineering Business Processes. In: Bordini, R.H., Dennis, L.A., Lesperance, Y. (eds.) Proc. of the 7th International Workshop on Engineering Multi-Agent Systems, EMAS 2019, held in conjuction with AAMAS 2019. Montreal, Canada (May 13-14 2019), http://cgi.csc.liv.ac.uk/ lad/emas2019/
5. Baldoni, M., Baroglio, C., Chopra, A.K., Singh, M.P.: Composing and Verifying Commitment-Based Multiagent Protocols. In: Wooldridge, M., Yang, Q. (eds.) Proc. of 24th International Joint Conference on Artificial Intelligence, IJCAI 2015. Buenos Aires, Argentina (July 25th-31th 2015), http://ijcai-15.org/
6. Baldoni, M., Baroglio, C., May, K.M., Micalizio, R., Tedeschi, S.: Computational Accountability in MAS Organizations with ADOPT. Applied Sciences **8**(4) (2018)
7. Boella, G., van der Torre, L.W.N., Verhagen, H.: Introduction to normative multiagent systems. In: Normative Multi-agent Systems. Dagstuhl Seminar Proceedings, vol. 07122 (2007)
8. Chopra, A.K., Singh, M.P.: From social machines to social protocols: Software engineering foundations for sociotechnical systems. In: Proc. of the 25th Int. Conf. on WWW (2016)
9. Coutinho, L.R., Sichman, J.S., Boissier, O.: Modelling dimensions for agent organizations. In: Handbook of research on multi-agent systems: Semantics and dynamics of organizational models, pp. 18–50. IGI Global (2009)
10. Cranefield, S., Oren, N., Vasconcelos, W.: Accountability for practical reasoning agents. In: AT 2018: 6th International Conference on Agreement Technologies. LNCS (2018)
11. Craven, R., Sergot, M.J.: Agent strands in the action language nc+. J. Applied Logic **6**(2), 172–191 (2008)
12. Fernandez, J.C., Mounier, L., Pachon, C.: A model-based approach for robustness testing. In: Proceedings of the 17th IFIP TC6/WG 6.1 International Conference on Testing of Communicating Systems. pp. 333–348. TestCom'05 (2005)
13. Grant, R.W., Keohane, R.O.: Accountability and Abuses of Power in World Politics. The American Political Science Review **99**(1) (2005)
14. Haller, P., Sommers, F.: Actors in Scala - concurrent programming for the multi-core era. Artima (2011)
15. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE. IJAOSE **1**(3/4), 370–395 (2007). https://doi.org/10.1504/IJAOSE.2007.016266, https://doi.org/10.1504/IJAOSE.2007.016266
16. Jones, A.J.I., Sergot, M.J.: A formal characterisation of institutionalised power. Logic Journal of the IGPL **4**(3), 427–443 (1996)
17. Jones, A.J., Carmo, J.: Deontic logic and contrary-to-duties. In: Gabbay, D. (ed.) Handbook of Philosophical Logic, pp. 203–279. Kluwer (2001)
18. Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Prentice Hall PTR, Upper Saddle River, NJ, USA (2004)
19. Sergot, M.J.: A computational theory of normative positions. ACM Trans. Comput. Log. **2**(4), 581–622 (2001)
20. Singh, M.P.: Distributed Enactment of Multiagent Workflows: Temporal Logic for Web Service Composition. In: The Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings. pp. 907–914. ACM (2003)
21. Therborn, G.: Back to norms! on the scope and dynamics of norms and normative action. Current Sociology **50**, 863–880 (2002)
22. Vincent, N.A.: Moral Responsibility, Library of Ethics and Applied Philosophy, vol. 27, chap. A Structured Taxonomy of Responsibility Concepts. Springer (2011)
23. Yolum, P., Singh, M.P.: Commitment Machines. In: Intelligent Agents VIII, 8th Int. WS, ATAL 2001. LNCS, vol. 2333, pp. 235–247. Springer (2002)