Demonstrating Exception Handling in JaCaMo

Matteo Baldoni[1], Cristina Baroglio[1], Olivier Boissier[2], Roberto Micalizio[1], and Stefano Tedeschi[1] (✉)

[1] Università degli Studi di Torino - Dipartimento di Informatica, Torino, Italy
firstname.lastname@unito.it
[2] Laboratoire Hubert Curien UMR CNRS 5516, Institut Henri Fayol, MINES Saint-Etienne, Saint-Etienne, France
Olivier.Boissier@emse.fr

Abstract. Current models for multi-agent organizations offer effective abstractions to build distributed systems, but lack a systematic way to address exceptions. This demonstration presents an exception handling mechanism in the context of the JaCaMo framework, based on the notions of responsibility and feedback. The mechanism is seamlessly integrated within organizational concepts, such as goals and norms.

Keywords: Exception Handling · Multi-Agent Organizations · JaCaMo

1 Introduction

System robustness is a main concern in software engineering practice. It is "the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions" [1]. It refers to the ability of a computer system to cope with *perturbations* occurring during execution, keeping an acceptable behavior [7]. *Exception handling* has been successfully proposed as a powerful technology to achieve robustness. When a perturbation is detected (e.g., missing parameters, or unknown data format) an *exception* breaks the normal flow of execution, and deviates it to a pre-registered *handler*, that is executed to manage the perturbation. Notably, the seminal work by Goodenough [8] points out how exceptions are a viable mechanism for structuring and modularizing software, separating concerns into components that interact. Exceptions permit to extend an operation's domain or its range. They tailor an operation's results to the purpose in using it, making them usable in a wider variety of contexts. The invoker of an operation controls the response to an exception to be activated; this increases the generality of an operation because the appropriate "fixup" will vary from one use to the next. Multi-Agent Systems (MAS), and organizations (MAOs) in particular, bring software modularization, and separation of concerns to an extreme. Key features of many organizational models (see e.g., [4–6]), are a functional decomposition of an organizational goal and a normative system. Norms shape the scope of the responsibilities that agents take within the organization, capturing what they should do to contribute to the organizational goal. Surprisingly, exception handling has almost never been applied in MAS as

postulated in [8]. A critical point is that the availability of *feedback* concerning the perturbation is crucial for treating exceptions, and hence for robustness [2, 3], but not easy to obtain in distributed systems. This demonstration shows how exception handling can be grafted inside the normative system of a MAO by explicitly distributing the responsiblities of rising and handling exceptions among agents, so making a perturbation feedback flow properly within the organization. We introduced exception handling in JaCaMo [4], integrating it at a conceptual level, within the abstractions of its meta-model, and at a software level, by enriching its infrastructure.

2 Main Purpose

The discussion above highlights two main aspects of exception handling: (i) it involves two parties: one is *responsible* for raising an exception, another is *responsible* for handling it, and (ii) it captures the need for some *feedback* from the former to the latter that allows coping with the exception. Since MAOs are built upon responsibilities, we claim that exception handling – in essence, a matter of responsibility distribution – can be integrated seamlessly. We interpret an exception as an event which denotes the impossibility, for some agent, to fulfill its responsibilities – e.g., a failure in goal achievement – causing the suspension of the distributed execution. We propose to leverage responsibility not only to model the duties of the agents in relation to the organizational goal, but also to enable them report about exceptions, occurring within the organization operation, and to identify the ones entitled for handling them. When agents join an organization, they will be asked to take on the responsibilities not only for organizational goals, but also: (i) for providing feedback about the context where exceptions are detected while pursuing goals, and (ii) if appointed, for handling such exceptions once the needed information is available. In our perspective, responsibilities also define the scope of exceptions, expressed with respect to the state of the organization, that agents ought to raise or treat. As a result, the normative system enables the coordination of the activities that concern not only the normal behavior of the system, but also its exceptional one, uniformly.

3 Demonstration

JaCaMo [4] is a platform that integrates agents, environments and organizations. Its organization model comprises a *structural* dimension, a *functional* dimension, including a set of schemes that capture how the organizational goals are decomposed into subgoals, grouped into missions, and a *normative* dimension binding the other two. Agents are held to commit to missions, taking responsibility for mission goals. We enriched the scheme specification with a few new concepts [3]. Shortly, **Recovery Strategy** enables the treatment of an exception by binding a **Notification Policy** with one, or many, **Handling Policy**. **Notification**

[3] The extension is available at http://di.unito.it/moiseexceptions.

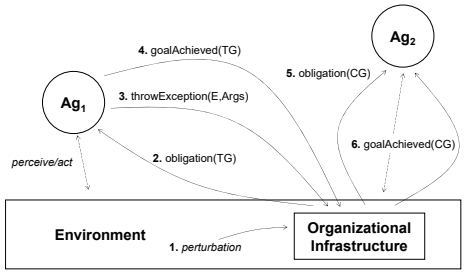

Fig. 1. Interaction between agents and organization for exception handling.

Policy specifies (i) when an exception must be raised by means of a **Throwing Goal**, enabled when a given perturbation occurs, and (ii) the kind of feedback to produce via **Exception Spec**. **Handling Policy**, instead, specifies how an exception must be handled, once the needed information–the exception feedback–is available, by enabling an appropriate **Catching Goal**. Notably, throwing and catching goals specialize the JaCaMo **Goal** concept, and are incorporated into mission like standard ones. In this way, our extension is seamlessly integrated into the organization management infrastructure, enabling a uniform approach in agent programming. Agents, entering an organization, take some responsibilities by committing to missions. Then, the infrastructure issues *obligations* to achieve goals, which may concern the normal functioning or the raising/handling of exceptions. Agent developers are required to implement the set of plans to make agents able to fulfill their responsibilities, achieving the goals expressed by the obligations directed to them. Figure 1 shows the typical interaction schema, between the involved agents and the organizational infrastructure, for handling the occurrence of an exception. As soon as a perturbation is detected, a suitable recovery strategy is searched and its notification policy activated: an obligation to achieve the corresponding throwing goal `TG` is issued. The agent responsible for `TG` will fulfill the obligation by throwing an exception (i.e., providing feedback) compliant with the specification. This enables one (or more) handling policy(ies), and the obligation(s) for the related catching goal(s) `CG` is issued. The agent in charge of `CG` leverages the exception feedback, made available through the infrastructure, to actually handling the exception. During the demonstration, we will present a set of use cases that take inspiration from real-world scenarios – coming from the fields of, e.g., business processes, and smart factories.

4 Conclusions

We evaluated the proposal with respect to a set of features that, in our view, should be exhibited by an exception handling mechanism to be suitable for MAS.

Autonomy Preservation. Exception handling should not interfere with the agents' autonomy. In our proposal, an explicit responsibility assumption creates expectation on the agents' behavior w.r.t. exception handling. Nonetheless,

agents remain free to violate their obligations. Agents are also autonomous in deliberating the most suitable way to carry out exception handling.

Decentralization. The mechanism leverages the distributed nature of MAS in the exception handling process. Exceptions are raised and handled in synergy by the society of agents in the organization. At the same time, the exception handling mechanism is integrated within the organizational infrastructure, which is reified in the environment in a distributed way.

Responsibility Distribution. Following Goodenough, exception handling is a mechanism to widen the scope of operations. Our proposal aims at creating a bridge between the agents responsible for raising exceptions and the ones responsible for their handling. The responsibility for exception handling is then moved outside the failing agent, increasing the generality of the applied recovery.

Importance of Feedback. A feedback coming from an informed source allows to increase the situational awareness about a perturbation, with straightforward benefits in its handling. This is especially true in a multi-agent setting, where each agent may have a different and partial view of the environment and of the overall ongoing execution. Our proposal systematize the way in which this relevant information is produced, encoded, delivered, and exploited for recovery.

Platform Integration. Together with the conceptual and theoretical soundness, we believe that the presence of a concrete programming support is fundamental for any application of exception handling. We then decided to implement the proposed model in the context of JaCaMo. The resulting solution proved to be effective in dealing with a wide range of situations, in multiple applications.

Acknowledgements Stefano Tedeschi's research project has been carried out thanks to the grant "Bando Talenti della Società Civile" promoted by Fondazione CRT with Fondazione Giovanni Goria.

References

1. ISO/IEC/IEEE International Standard - Systems and software engineering – Vocabulary. ISO/IEC/IEEE 24765:2010(E) (2010)
2. Alderson, D.L., Doyle, J.C.: Contrasting views of complexity and their implications for network-centric infrastructures. IEEE Tr. on Sys., Man, and Cyber. **40**(4) (2010)
3. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Robustness based on accountability in multiagent organizations. In: AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems. pp. 142–150. ACM (2021)
4. 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)
5. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. JAAMAS **8**(3), 203–236 (2004)
6. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20**(1), 3 – 50 (1993)
7. Fernandez, J.C., Mounier, L., Pachon, C.: A model-based approach for robustness testing. In: Proc. of TestCom 2005. pp. 333–348 (2005)
8. Goodenough, J.B.: Exception handling: Issues and a proposed notation. Communications of the ACM **18**(12), 683–696 (1975)