Exceptions and Accountability for Robust Applications of JaCaMo

EUMAS 2024 – Agent Toolkits Track

Matteo Baldoni[1], Cristina Baroglio[1], Roberto Micalizio[1], Stefano Tedeschi[2]

August 28, 2024 - The 21st European Conference on Multi-Agent Systems August 26-28th, 2024, Dublin, Ireland

[1]Università di Torino, Torino, Italy
[2]Università della Valle d'Aosta - Université de la Vallée d'Aoste, Aosta, Italy

Motivation

- Robustness is a crucial requirement of modern, distributed software systems [Jain et al., 1999, Kalia and Singh, 2015, Christie et al., 2021, Christie et al., 2022]
- Multi-Agent Systems are an effective approach to realize distributed systems by means of heterogeneous and autonomous agents
- Multi-agent *organizations* (MAOs), in particular, provide useful abstractions for modularizing code spread over many components, and orchestrate their execution by way of norms
  - JaCaMo [Boissier et al., 2020] is one of the best-known platforms for implementing multi-agent organizations

Motivation

- Unfortunately, JaCaMo focuses on providing the means for capturing the normal, correct behavior of the system only
- It lacks of structural mechanisms allowing agents to exchange and propagate information (*feedback*) when they face perturbations, thereby supporting robustness, a crucial feature to build robust distributed systems [Alderson and Doyle, 2010]
- Agents are peers (nor related by relationships like caller-callee, parent-child): the realization of robustness should occur through the definition of appropriate distributions of responsibilities among agents

JaCaMo+Exception+Accountability: our proposal

- We present two extensions to the JaCaMo platform:
  - **JaCaMo+Exception**:
    Exception handling is suitable for treating exception anticipated at design time by activating handlers, that are also specified at design time
  - **JaCaMo+Accountability**:
    Accountability is suitable for defining structured "channels" that agents can use at runtime to gain situational awareness about the situation of interest and then take actions.
- Raising and handling exceptions, asking, returning, and treating an account are tasks/goals under the responsibility of specific agents

JaCaMo+Exception+Accountability: background

- JaCaMo integrate three different programming platforms: agents (Jason [Bordini et al., 2007]), environments (CArtAgO [Ricci et al., 2009]), and organizations (Moise [Hübner et al., 2007])

JaCaMo+Exception+Accountability: background

- The organizational model has three dimensions:
  - The **structural dimension**: it specifies roles, groups, and links between roles in the organizations
  - The **functional dimension**: it encompasses one or more schemas that elicit how the organizational goals are decomposed into sub-goals and how these sub-goals are grouped in coherent missions
  - The **normative dimension**: it distributes the responsibilities to roles, by specifying role permissions and obligations for missions
- The organizational infrastructure is part of the environment through a set of dedicated organizational artifacts
- The organizational specification is translated into a normative program (NOPL [Hübner et al., 2009]), which is interpreted by the organizational artifacts

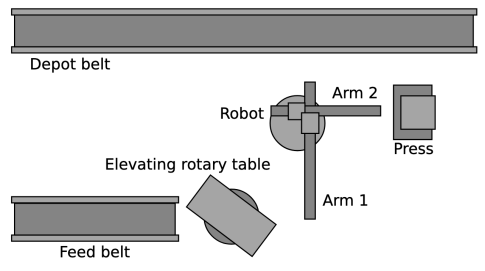
JaCaMo+Exception+Accountability: platform integration

- In order to include exception handling and accountability, we worked at three levels:
  - **Specificaton level**: We enriched the specification of an organization with the specification of a set of *notification policies* (for exceptions) and *accountability agreements*
  - **Normative level**: We extended the normative program in NOPL to issue the obligations related to the notification policies and accountability agreements
  - **Infrastructure level**: We enriched the organizational infrastructure with the functionalities needed by agent to raise or handle exceptions and to provide or treat accounts

JaCaMo+Exception+Accountability: Specification level

We extended the specification level with:

- **Notification policies** to specify when an exception occurs, how notify it (*raising goal*), and how handling it (*handling goal*)
- **Accountability agreements** to specify when an account may be requested (*requesting goal*), how to report it (*accounting goal*), and how treat it (*treatement goal*)

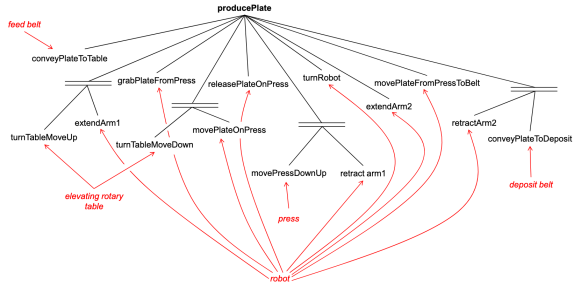


A production cell for metal plates inspired to [Lewerentz and Lindner, 1995]

JaCaMo+Exception+Accountability: Specification level

We extended the specification level with:

- **Notification policies** to specify when an exception occurs, how notify it (*raising goal*), and how handling it (*handling goal*)
- **Accountability agreements** to specify when an account may be requested (*requesting goal*), how to report it (*accounting goal*), and how treat it (*treatement goal*)



Scheme of the functional decomposition in JaCaMo

JaCaMo+Exception+Accountability: Specification level

We extended the specification level with:

- **Notification policies** to specify when an exception occurs, how notify it (*raising goal*), and how handling it (*handling goal*)
- **Accountability agreements** to specify when an account may be requested (*requesting goal*), how to report it (*accounting goal*), and how treat it (*treatement goal*)

```
<notification-policy id="npMotor" target="turnTableMoveUp"
        condition="scheme_id(S) &amp;amp; failed(S,turnTableMoveUp)">
    <exception-specification id="exMotor">
        <exception-argument id="motorNumber" arity="1" />
        <raising-goal id="notifyStoppedMotorNumber" />
        <handling-goal id="motorFix">
            <plan operator="sequence">
                <goal id="scheduleTableMotorFix" />
                <goal id="pauseProduction" />
            </plan>
        </handling-goal>
    </exception-specification>
</notification-policy>
```

Notification policy for exception handling

JaCaMo+Exception+Accountability: Specification level

We extended the specification level with:

- **Notification policies** to specify when an exception occurs, how notify it (*raising goal*), and how handling it (*handling goal*)
- **Accountability agreements** to specify when an account may be requested (*requesting goal*), how to report it (*accounting goal*), and how treat it (*treatement goal*)

```
<accountability-agreement id="aa1" target="conveyPlateToTable" condition="true">
    <account-template id="stock">
        <account-argument id="availablePlates" arity="1" />
        <requesting-goal id="requestRemainingStock" />
        <accounting-goal id="notifyRemainingStock" />
        <treatment-goal id="slowDownProduction"
            when="account(stock,_,[availablePlates(N)]) &amp;
                ( N &lt;= 10 &amp; N &gt; 0 )" />
        <treatment-goal id="stopProduction"
            when="account(stock,_,[availablePlates(0)]) " />
    </account-template>
</accountability-agreement>
```

Accountability agreement for accounting

JaCaMo+Exception+Accountability: Agent level

We extended the agent level with:

- **goalFailed(G)**, by which an agent can signal the failure of a goal
- **raiseException(E, Args)**, by which an agent can raise an exception with a list of arguments (template)
- **giveAccount(A, Args)**, by which an agent can report an account with a list of possible arguments (template), upon request
- **goalReleased(G)**, by which an agent can release a failed (organizatinal) goal

These are used in a Jason agent program

JaCaMo+Exception+Accountability: Availability

- It is available here: `https://di.unito.it/moiseexceptions`
- It is available here: `https://di.unito.it/moiseaccountability`

Indeed, very soon in a branch of the official Moise distribution in a single extension!

More information in
[Baldoni et al., 2021a, Baldoni et al., 2021b, Baldoni et al., 2022, Tedeschi, 2021] for exception handling and in
[Baldoni et al., 2021c, Baldoni et al., 2021d, Baldoni et al., 2023] for accountability

Alderson, D. L. and Doyle, J. C. (2010).
Contrasting views of complexity and their implications for network-centric infrastructures.
IEEE Tr. on Sys., Man, and Cyber., 40(4).

Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., and Tedeschi, S. (2021a).
Demonstrating Exception Handling in JaCaMo.
In Dignum, F., Corchado, J. M., and De La Prieta, F., editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection - 19th International Conference, PAAMS 2021, Salamanca, Spain, October 6-8, 2021, Proceedings*, volume 12946 of *Lecture Notes in Computer Science*, pages 341–345. Springer.

Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., and Tedeschi, S. (2021b).
Distributing Responsibilities for Exception Handling in JaCaMo.

In Endriss, U., Nowé, A., Dignum, F., and Lomuscio, A., editors, *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pages 1752–1754. International Foundation for Autonomous Agents and Multiagent Systems.


Baldoni, M., Baroglio, C., Micalizio, R., and Tedeschi, S. (2021c).
Reimagining Robust Distributed Systems through Accountable MAS.
IEEE Internet Computing, 25(6).


Baldoni, M., Baroglio, C., Micalizio, R., and Tedeschi, S. (2021d).
Robustness Based on Accountability in Multiagent Organizations.
In Endriss, U., Nowé, A., Dignum, F., and Lomuscio, A., editors, *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pages 142–150. International Foundation for Autonomous Agents and Multiagent Systems.

Baldoni, M., Baroglio, C., Micalizio, R., and Tedeschi, S. (2022).
Exception handling as a social concern.
IEEE Internet Computing, 26(6):33–40.

Baldoni, M., Baroglio, C., Micalizio, R., and Tedeschi, S. (2023).
Accountability in multi-agent organizations: from conceptual design to agent programming.
Autonomous Agents and Multi-Agent Systems, 37(1):1–37.

Boissier, O., Bordini, R. H., Hübner, J., and Ricci, A. (2020).
Multi-agent oriented programming: programming multi-agent systems using JaCaMo.
MIT Press.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007).
Programming Multi-Agent Systems in AgentSpeak Using Jason.

John Wiley & Sons.


Christie, S. H., Chopra, A. K., and Singh, M. P. (2021).
Bungie: Improving fault tolerance via extensible application-level protocols.
Computer, 54(5):44–53.


Christie, S. H., Chopra, A. K., and Singh, M. P. (2022).
Mandrake: multiagent systems as a basis for programming fault-tolerant decentralized applications.
Autonomous Agents and Multi-Agent Systems, 36(1).


Hübner, J. F., Boissier, O., and Bordini, R. H. (2009).
A normative organisation programming language for organisation management infrastructures.
In *Coordination, Organizations, Institutions and Norms in Agent Systems V*, volume 6069 of *Lecture Notes in Computer Science*, pages 114–129. Springer.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2007).
Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels.
Int. J. Agent-Oriented Softw. Eng., 1(3/4):370–395.

Jain, A. K., Aparico IV, M., and Singh, M. P. (1999).
Agents for process coherence in virtual enterprises.
Communications of the ACM, 42(3):62–69.

Kalia, A. K. and Singh, M. P. (2015).
Muon: designing multiagent communication protocols from interaction scenarios.
Autonomous Agents and Multi-Agent Systems, 29(4):621–657.

Lewerentz, C. and Lindner, T. (1995).

Case study "production cell": A comparative study in formal specification and verification, pages 388–416.
Springer.


Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009).
Environment Programming in CArtAgO, pages 259–288.
Springer US, Boston, MA.


Tedeschi, S. (2021).
Exception Handling for Robust Multi-Agent Systems.
PhD thesis, Università degli Studi di Torino, Dipartimento di Informatica, Torino, Italy.