Chapter 2
Multi-Agent System Properties

Abstract This chapter succinctly presents Multi-Agent Systems (MAS) as software architecture composed of entities: agents, presenting specific properties: autonomy and pro-activeness; and a platform providing agents for communication means. These properties are then analyzed with regard to their impact on possible faults and impacts on system dependability.

Keywords Multi-agent system · Autonomy · Pro-activeness · Adaptability · Dependability · Non deterministic behavior

2.1 Multi-Agent System and Agents

A multi-agent system is a set of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each individual agent. We call a "platform" whatever allows those agents to interact not taking into consideration the shape that such a platform can take (centralized or not, embedded into the agents or not, …). This platform usually provides agents with a set of services depending on the system needs and is considered as a tool for the agents, it does not exhibit an autonomous or pro-active behavior.

Agents, according to MAS community (see for example [1]) have the following properties:

- *Autonomy*. An agent possesses individual goals, resources and competences; as such it operates without direct human or other intervention, and has some degree of control over its actions and its internal state. One of the foremost consequences of agent autonomy is agent adaptability as an agent has the control over its own state and so can regulate its own functioning without outside assistance or supervision.
- *Sociability*. An agent can interact with other agents, and possibly humans, via some kind of agent communication language. Through this means, an agent is able to provide and ask for services.

K. Potiron et al., *From Fault Classification to Fault Tolerance for Multi-Agent Systems*, SpringerBriefs in Computer Science, DOI: 10.1007/978-1-4471-5046-6_2,

- *Reactivity*. An agent perceives and acts, to some degree, on its close environment; it can respond in a timely fashion to changes that occur around it.
- *Pro-activeness*. Although some agents, called reactive agents, will simply act in response to stimulations from their environment, an agent may be able to exhibit goal-directed behavior by taking the initiative.

Even if we will not address the subject of defining the autonomy, we will study the consequences of a point that most definitions have in common. The point is that autonomy allows agents to take their decisions on their own, e.g. "without direct human or other external intervention".

For a commonly accepted definition of agent autonomy, we would like to mention Henry Hexmoor's [2]:

"An agent is autonomous with respect to another agent, if it is beyond the influences of control and power of that agent."

As well as the definitions given in [3, 4].

These definitions of agent autonomy implies that agents are not entitled to dictate others decisions or actions. Those are huge hypothesis for the agents that can lead to an unproductive system if no agent would like to work with others and must be balanced with the collaborative aspects of agents. This can be seen as: "I will work with you, not because you have the power the force me to but because of mutual interest and if I have a more profitable offer I will stop working with you".

Moreover agents do not know the decision criteria of the other agents neither during execution (agents are distributed and partitioned entities and so have no access to internal variables of others) nor than during conception (interactions are defined but the intended behavior of the other agents are not known). Those are every day condition when building a software system; more and more entities of the system are not built by the same team or company (and in some cases not in the same country).

As a consequence, agents gain some independence with regard to the other agents; they can go on and thus survive even if no other agent is available.

This aspect of autonomy, decision criteria are internal and private to the agent, provide agent with more independence with regard to the other agents (they can go on and survive even if other agents are unavailable. This contributes to system robustness through the modularity and fault isolation aspects.

Nonetheless, autonomy makes the behavior of agents not completely foreseeable and therefore non-deterministic. This unpredictability is a property of interacting entities; therefore, agents taking "their decisions by themselves" (i.e. by taking non-supervised decisions on partial knowledge[1]) can, whether voluntarily or not, be responsible for faults that other agents will experience.

[1] Note that the "partial knowledge" consideration is directly a part of *Reactivity* definition given by [1].

This faulty aspect of agent autonomy had also been underlined by Hägg in [5]:

"When discussing fault handling in multi-agent systems (MAS) there are often two opposite positions. The first says that a MAS is inherently fault tolerant by its modular nature; faults can be isolated and generally do not spread. The other says that a MAS is inherently insecure, as control is distributed. It is non-deterministic, and a specific behavior is hard to guarantee, especially in fault situations."

This distributed way of controlling the application induces the fact that agents undergo the autonomous behavior of other agents in two different ways:

- One agent may choose (using its internal methods and private variables) not to respond to a request from another agent for any reason it considers acceptable (suspicion toward the other agent, more important things to do, schedule overlap, being overloaded…);
- Malicious agents can choose to harm other agents. From a software engineering point of view, autonomy can be perceived as the fact that the agent designer does not know exactly the specifications, internal laws or internal state of other agents during their execution. Even if MAS are distributed systems, autonomy is the breaking point. Moreover, in some MAS (called open MAS), even the developers (and owners) of the other agents may not be known.

Note that form the "game" point of view such a malicious and harming agent may not be a *faulty* agent but a *winning* agent, if harming other agents allows itself to obtain a benefit. In a competition environment, autonomy and pro-activeness of agents may be expressed in various ways.

2.2 Agents and Faults

As agents know that other agents are autonomous and then unpredictable, they must take their decisions on their own and be able to adapt themselves to changing situations. Doing so, they gain independence or a degree of freedom with regard to their relations to other agents. They can go on and thus survive even if other agents are not available. This aspect of autonomy facilitates two aspects of fault tolerance:

- Error confinement, since the isolation of one or more suspected faulty agent(s) may not result in totally incapacitating the other agents and,
- System readjustment, since agents considering that others are autonomous must have adaptability capacities that result in the fact that a loss of some agents will not imply a complete loss of service for the MAS.

Those two aspects make community of agents (and so Multi-Agent Systems) more robust.

But at the same time, it is also said that autonomy makes fault tolerance a more difficult task—see [5, 6]—and one can wonder why such a statement was expressed.

This statement is a direct consequence of the decision-making process resulting from agent autonomy: the behavior of an agent is not completely foreseeable for the agents interacting with it, which requires for the agent to be prepared to adapt its behavior to unforeseen situations. Considering that this adaptation is needed for every decision and action of other agents results in the creation of a lot of unpredictable situations that should be handled. This aspect does not facilitate agents design.

It is then clear that agents cannot be designed like classical systems and fault tolerance must have a particular place in their design. As a matter of fact, since autonomy and pro-activeness are a natural feature of agents, faults induced by their behavior cannot be "removed" like it may be the case for development faults in monolithic deterministic software. This became even more complicated when emerging behaviors are taken into consideration.

Autonomy, *limited environment sensing* and *adaptability* are interesting properties since they correspond to the way software development is evolving and make multi-agent systems relevant even in industrial or military domains like airborne systems.

Software systems being more and more complex imply that:

- Safety studies cannot predict every possible harmful behavior of the system.
- Software modules of a system cannot be aware of all actions of other modules.

Then reconfiguration is one of the few adapted solution for the system to be able to have a better continuity of service.

This book tries to answer the questions rising from these considerations and to provide multi-agents system with new arguments for their use in industrial or military domains:

- What is to be done to specify and implement an agent interacting with such unpredictable entities?
- What is to be done to specify and implement the whole MAS?

2.3 Agents and Fault Tolerance

Some work on MAS has been done with regard to fault tolerance domain that provides agents and MAS designers with responses to these questions.

Two main line of research in the MAS domain are addressing fault tolerance (we exclude here the ad hoc adaptation of interaction protocols and non reusable methods), they are presented next.

One line of research for agents fault tolerance is to include the fault tolerance directly into the languages representing the behavior of the agent. The creators of Cool [7] or AgenTalk [8] in their chapters introducing their respective languages also present some prospects on mechanisms to handle faults into the conversations.

They propose to use a meta script to control the execution of the behavioral scripts, the meta scripts being started with messages, representing the exceptions, sent from the behavioral script to the meta script. The authors do not provide any general meta script of control and assume that they will be built in a suitable way for each exception. They conclude saying that the meta scripts may be grouped together using exception classes.

More recently, Dragoni and Gaspari [9] proposed a fault tolerant agent communication language (FT-ACL) that deals with crash failure of agent, providing fault tolerant communication primitives and support for an anonymous interaction protocol. The knowledge level of the FT-ACL is based on giving for each speech act a success continuation and a failure continuation.

These methods are focused on conversational agents and on the handling of faults perceived into the communications but the realization of the suitable handler is the responsibility and the burden of the agent designer/developer. The more serious issue of such a handler creation is the risk for the developer to introduce new faults into its handler.

Another line of research in MAS, on exception handling for MAS uses sentinels [5, 10–12] to diagnose (from the analysis of the received and sent messages) and handle exceptions for agents. This approach is different from the approaches presented previously since the diagnosis and decision about the handler adapted to the fault is outside the agent. The choice of the handler can be done inside a knowledge-base containing specific handlers [10]. Besides the creation of the external entity, it requires some special design of the agents to allow an external entity to change its internal state.

And there are also other lines of research in MAS fault tolerance like for example:

- The replication (or cloning) of agents is used in [13, 14] to deal with physical faults.
- "Prevention of harmful behaviors" [15] which deals with the emergence of harmful behaviors of agents.

Several lines of research in MAS never refer to fault tolerance but provide valuable strategies for it, like:

- Computational trust and reputation [16] to deal with malicious agents,
- Agent planning [17, 18] to deal with unexpected situations.

In all these research works, it appears that the kind of faults taken into consideration is never clear and hence the efficiency of the proposed fault tolerance solutions with regard to fault handling is difficult to estimate. A way to address these questions is to study faults in order to classify them in a limited number of fault classes and this is indeed what will be presented in the next chapter.

References

1. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theories, Architectures and Languages, January 1995. Lecture Notes in Artificial Intelligence, vol. 890, ISBN 3-540-58855-8
2. Hexmoor, H.: Stages of autonomy determination. Trans. Sys. Man Cyber. Part C **31**(4), 509–517 (2001)
3. d'Inverno, M., Luck, M.: Understanding autonomous interaction. In: Wahlster, W. (ed.) Proceedings of the 12th European Conference on Artificial Intelligence, pp. 529–533. John Wiley and Sons (1996)
4. Castelfranchi, C., Falcone, R.: From automaticity to autonomy: the frontier of artificial agents. In: Hexmoore, H., Castelfranchi, C., Falcone, R. (eds.) Agent Autonomy, Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 7, pp. 103–136. K. Academic (2003)
5. Hägg, S.: A sentinel approach to fault handling in multi-agent systems. Paper presented at Second Australian Workshop on Distributed AI in conjunction with the Fourth Pacific Rim International Conference on Artificial Intelligence, pp. 181–195. Springer-Verlag, London (1996)
6. Zhang, Y., Manisterski, E., Kraus, S., Subrahmanian, V.S., Peleg, D.: Computing the fault tolerance of multi-agent deployment. Artif. Intell. **173**(3–4), 437–465 (2009)
7. Barbuceanu, M., Fox, M.S.: Cool: a language for describing coordination in multiagent systems. In: Lesser, V., Gasser, L. (eds.), Proceedings of the First International Conference oil Multi-Agent Systems, June 12–14, 1995, San Francisco, California, USA, pp. 17–24, The MIT Press (1995)
8. Koren, I., Koren, Z., Su, S.Y.: Analysis of a class of recovery procedures. IEEE Trans. Comput. **35**(8), 703–712 (1986)
9. Dragoni, N., Gaspari, M.: Crash failure detection in asynchronous agent communication languages. Auton. Agent. Multi-Agent Syst. **13**(3), 355–390 (2006)
10. Klein, M., Dellarocas, C.: Exception handling in agent systems. In: Etzioni, O., Müller, J.P., Bradshaw, J.M. (eds.) Proceedings of the Third International Conference on Autonomous Agents (Agents'99), ACM Press, Seattle, pp. 62–68 (1999)
11. Platon, E., Sabouret, N., Honiden, S.: A definition of exceptions in agent oriented computing. In: O'Hare, G., O'Grady, M., Dikenelli, O. Ricci, A. (eds.) Proceedings of the 7th International Conference on Engineering Societies in the Agents World VII, pp. 161–174, Springer-Verlag, Berlin, Heidelberg (2007)
12. Shah, N., Chao, K.-M., Godwin, N., James, A.E.: Exception diagnosis in open multi-agent systems. In: IAT, pp. 483–486 (2005)
13. Fedoruk, A., Deters, R.: Improving fault-tolerance by replicating agents. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: part 2, ACM Press, Bologna, Italy, pp. 737–744. (2002)
14. Guessoum, Z., Faci, N., Briot, J.-P.: Adaptive replication of large-scale multiagent systems: towards a fault-tolerant multi-agent platform. In: Proceedings of the Fourth International Workshop on Software engineering for Lrge-Scale Multi-Agent Systems, ACM Press, St. Louis, Missouri, pp. 1–6. (2005)
15. Chopinaud, C., Fallah-Seghrouchni, A.E., Taillibert, P.: Prevention of harmful behaviors within cognitive and autonomous agents. Paper presented at European Conference on Artificial Intelligence, pp. 205–209 (2006)
16. Sabater, J., Sierra, C.: Review on computational trust and reputation models Artif. Intell. Rev. **24**(1), 33–60 (2005)
17. de Weerdt, M., ter Mors, A., Witteveen, C.: Multi-agent planning: an introduction to planning and coordination. In: Handouts of the European Agent Summer School, pp. 1–32, (2005)
18. Seghrouchni, A.E.F., Hashmi, M.A.: Multi-agent planning. In: Essaaidi M. et al. (eds.) Software Agents, Agent Systems and Their Applications. IOS Press (NATO Science for Peace and Security Series) (2012)

http://www.springer.com/978-1-4471-5045-9

From Fault Classification to Fault Tolerance for Multi-Agent Systems

Potiron, K.; El Fallah Seghrouchni, A.; Taillibert, P.

2013, VIII, 80 p. 19 illus., Softcover
ISBN: 978-1-4471-5045-9