Agent Design from the Autonomy Perspective

Massimo Cossentino[1] and Franco Zambonelli[2]

[1] Istituto di Calcolo e Reti ad Alte Prestazioni
Italian National Research Council
Viale delle Scienze Ed. 11
90128 Palermo, Italy
cossentino@pa.icar.cnr.it
[2] Dipartimento di Scienze e Metodi dell'Ingegneria
Universita di Modena e Reggio Emilia,
42100 Reggio Emilia, Italy
franco.zambonelli@unimore.it

Abstract. The design and development of multiagent systems can take advantage of a '*multi-perspectives*' approach to system design, separately focusing and the design and evaluation of one (or of a few) specific features of the system-to-be. In this paper, we introduce the basic concepts underlying the multi-perspectives approach. Then, we take a specific look at agent autonomy and try sketch to a new specific perspective to deal with it.

1 Introduction

Despite of the today already demonstrated advantages of multiagent systems (MASs) and of an agent-oriented approach to software development [18, 15, 10, 12], software developers must be aware that the design of a MAS tends to be more articulated than the design of a traditional object-oriented system. In fact, in the design of a MAS, one should take into account novel issues such as [3]: the proactive and reactive behavior of autonomous agents; dealing with social and ontological aspects of inter-agent communications; controlling dynamic interaction with an physical or computational environment; understanding the trust and security problems connected to a potentially opened system.

During the design of a MAS, the designer is forced to pass through several different levels of abstraction looking at the problem from many different points of view. The result is a series of models that reflect this multi-faced structure. As a natural consequence, the process used in affording the different models should not be linear but multi-dimensional. In other words, the most natural way of tackling the complexity of developing and representing MASs, and taking an effective advantage of agent-based paradigm, consists in a '*multi-perspectives*' approach to the system design [14, 5].

Such an approach implies representing the system-to-be according to several different '*perspectives*'; each one of them promoting an abstract representation of the system, and enabling the evaluation of one or a few features of the system,

thus highlighting some features and aspects of current interest, while hiding some others that are not interesting from that specific point of view.

In a previous paper [5] we already presented some perspectives that could effectively support and reduce the complexity of a MAS design process. In this paper, after having better introduced the concepts underlying the '*multi-perspectives*' approach to the system design, we take a specific look at agent autonomy. In particular, we will try to outline the characteristics of a new specific perspective explicitly conceived to deal with agent autonomy and related issues.

2 Different Perspectives for MAS design

Other works already exist where agents and MASs have been looked at from a multi-level or multi-perspective point of view.

In [5], we discussed five different perspectives in analyzing the system design. Two of them (knowledge and computer) come from the Newell's classification [14], later expanded by Jennings [11] with the inclusion of the social level. Two further perspectives (architectural and resource) come from classical software engineering concepts. However, in our opinion, looking at a MAS from different perspectives should not simply result in a series of different not-related subsystems, or in a partial descriptions of the whole system. Rather, the real outcome should be a more detailed description of the system in terms of a well-defined aspect.

When the designer looks at the system in order to study some specific problem, (s)he thinks about it as 'something' of specific. He can be concerned about the distribution of the software in the available hardware platforms in order to optimize the performances, or can be interested in defining the rules of interaction of the agent society. For this reason we intentionally call these different points of view 'perspectives' and not 'levels' or 'abstractions' because we want to stress the concept that the perspective is the representation of the system when the spectator is interested only in a specific conceptual area of the multi-faced agent system.

In order to better express our concepts we will now briefly present the key issues of some perspectives, followed by a discussion on the 'structure' that is behind a perspective. This will serve as introduction to the section where the autonomy perspective is presented and analyzed.

2.1 Review of Some Perspectives

To clarify our concepts, let us briefly sketch a few known perspectives that can be of use in MAS design.

The *architectural perspective* looks at the software as a set of functionalities to be implemented in a classical, software engineering approach [1]. This is clearly quite an abstract perspective: its elements exist in the mind of the designer,

since they are abstractions of the system representing the functionalities and their logical implementation.

The *social perspective* is characteristics – although not exclusive – of MASs. Many authors adopt a social or an organizational metaphor to describe MASs, and accordingly exploit some kind of social perspective in MAS design [13, 20]. Such perspective focuses on agents as individuals of a society (or members of an organization) that interact with each other to pursue social or selfish goals, as imposed by the designer.

The *knowledge perspective* is a highly detailed point of view. In such a perspective, the single agent and its functional and behavioral details (that induce some specific implementation), are conceived as entities that are able to manipulate some sort of external knowledge, with the goal of procuding/spreading new knowledge [14, 9].

The *resource perspective* is oriented toward the reuse of existing design or implementation resources (e.g. architectural patterns in software architectures or detailed design patterns for components and their tasks [8]). It represents agents/components/architectures in terms of some kind of mental bottom-up process, and deals with the process of recycling an existing agent (or parts of it) and eventually adapting it to match the necessity of a new problem.

The *computer perspective* is the more physical, touchable point of view. It relates to the spreading of files that constitute the software in the available hardware platforms and to the computational and storage load imposed by the processes in charge of executing the software system. This perspective therefore considers the deployment issues arising from the interplay between the hardware and the software system [14].

Depending on the specific application problem and on the specific characteristics of the MAS to be designed, only a portion of the above perspectives are likely to be of use. As it is common in software engineering problems, the designer should find the correct trade-off between the number of different descriptions of the system, the need to ensure their coherence and the consequent increasing number of concepts that (s)he has to manage. While introducing a new perspective could allow the identification and tracking of a potential risk in the system development, the uncontrolled use of this technique could produce the undesirable relapse of inducing the designer to consider so much variables that he could loose the perception of the system as a whole.

2.2 General Outline of a Perspective

In our work we refer to the concept of perspective instead of level also because we want to emphasize the unity of the system thought as a representation of the problem-solution couple that evolves from the early stages of the requirements elicitation to the final coding and deployment activities. The system can be represented in the sequence of models and phases of a design methodology with their resulting artifacts. Looking at this unit with different scopes we obtain a perspective of it that shows some elements (under one of their possible facades)

hiding what is out of the particular focus. In very general terms (and abstracting from the presence of agents and MASs) one can characterize a perspective as made of design *elements* that are composed abiding to some *constraints* in order to build a system conceived to operate in a specific *context*. The designer assembles these elements according to a (design) *rationale* that establishes the composition strategy by processing the *inputs* required by the perspective. Inputs, elements, context, rationale, and constraints are defined as follows:

1. *Inputs*. They defines the information that will be evaluated by the designer at design-time according to the prescribed design rationale. This will also be likely processed by the system (at run-time) in order to achieve its design objectives. Typically, these are static elements of the design either introduced and engineered by the designer or pre-existing in the environment. These inputs, belong to two different categories: goals/requirements/features of the system, and input data available for the system. Depending on the adopted perspective, the first type of inputs can be design goals, architectural concerns, cooperation/collaboration paradigms and so on. Input data (the second category) could be files, records, computational resources, or any type of abstract knowledge.
2. *Elements*. These are the elementary computational components of the perspective (e.g., depending on the perspective, these could be functions, agents, behaviors, software components, etc.). These elements are defined/refined by the designer acting according to the design rationale. Using them like bricks the designer will composes new pieces of the system completing the definition of its appearance from the specific perspective point of view.
3. *Context*. Each element of the perspective is thought to be applied in some operating scenarios (and bringing elements outside their natural operational context could cause a system failure). Environmental considerations affecting the system design can be enumerated among context concerns; for example we should consider specific characteristics or constraints of the environment that could influence the computational capabilities of the systems component.
4. *Rationale*. What the overall system will be obviously depends on the motivation underlying how each of the system element is composed with the others. Some design choices depend on specific strategies (for example respecting the holonic architecture) that can easily be formalized, others come from the designer skills and experience (it is a matter of fact that a system designed by a student is commonly less effective than the solution provided by a senior designer). In this work we will refer to the design rationale trying to include this untouchable contributions coming from experience and skill in form of guidelines. As a result, we think about the rationale always as a set of (someway) formalized rules, algorithms, conventions, best practices and guidelines that will guide the designer work. Of course, identifying and representing such a set of motivations strictly depends on the adopted perspective, on its inputs and elements.

5. *Constraints.* These defines the rules according to which the various elements of the system can be assembled for instance to compose a complex service or reach a global application goal. These rules are of course particularly important in all contexts where a complex service or a global application goal derives from the composition/interplay of the activities of the various components of the system, and where such a global goal can be obtained only by strictly respecting some composition rules. Also in this case, the specific adopted perspective influences the way in which these constraints are identified and expressed.

3 The Autonomy Perspective

A very distinguishing characteristic of MAS is their being composed of autonomous components, capable of proactive actions and of decisional capabilities. For this reason, and because the autonomy dimension in not something that 'traditional' software engineers are used to deal with, we think that the adoption of a specific autonomy perspective in MAS design may be needed.

Here we will try to sketch what an autonomy perspective in MAS could look like, by discussing it (according to the characterization of the Subsection 2.2) in terms of inputs, elements, context, rationale and constraints.

3.1 Inputs

The presence of some initial hypothesis, '*inputs*', is a common element of all the design activities. Such inputs will help the designer to devise and represent an appropriate architecture for the system-to-be. When adopting a specific perspective to design, of course, a limited set of '*inputs*' will be of interests.

In an autonomy perspective, whose focus in on analyzing a system from the viewpoint of the autonomous capabilities of a number of proactive, task-oriented and decision-making components, the design inputs of most interest are: '*system goals*' and '*domain ontology*'. They correspond, under the autonomy point of view, to the two fundamental needs of each perspective, i.e., the functionalities affecting the achievement of the design objectives in the specific perspective and the data to be processed by the system, respectively. In fact, the analysis of system goals guides the actual design activity in identifying the basic elements (i.e., autonomous goal-oriented agents) of the system-to-be. The analysis of the domain ontology helps identifying what knowledge will be available to system elements for them to achieve their functionalities (i.e., their goals).

While a variety of other '*inputs*' can be available to the designer of a MAS (e.g., specific non-functional requirements or specific models of knowledge acquisition) this will not play a role in an autonomy perspective, and have to be taken into account in other specific perspective (e.g., a computer perspective or a knowledge perspective).

3.2 Elements

Clearly, the elements of interests in an autonomy perspective for MAS design are autonomous agents, intended as proactive decision-making components combining specific proactive abilities (behaviors/roles) with available knowledge in order to reach goals inspired by their vocation.

The factors that mostly affect the autonomous behaviours of an agents and that should be taken into when modeling agents from the autonomy perspective are:

- Agent's vocation
- Agent's knowledge
- Agent's (behavioral) abilities
- Available resources

The agent's vocation is characterized by two interesting aspects (external and internal). The first one descends from the agent creator (the designer) point of view and it addresses the reason for which the agent has been created. This has a direct influence on the other aspect (the internal one): the will and consciousness that is put in the agent itself; in a BDI agent this could correspond to agent's desires, while in the PASSI approach [6] this is the requirement(s) that has to be fulfilled by the agent.

Agent's knowledge (at least some specific portion of it) is one of the elements of autonomy perspective since it contributes to agent autonomy by building up the consciousness the agent has of its operational scenario, and in most cases the strategy it will initially adopt is an a-priori one that is updated when new information about the situation will be available.

The expected result of the autonomous agent design is a combination of the agent's abilities to achieve some goal. Usually this is obtained with the correct coordination of some agent's behaviors and their specific duties (in terms of knowledge to be processed, options to be selected and so on). The analysis of the different agents (behavioral) capabilities is therefore one of the desiderata for the computational autonomy perspective.

The availability of specific type or resources (whether computational or physical) is another factor that influences the autonomous action of an agent. An agent could decide to adopt an alternative plan according to the possibility of using some kind of resource. Not all the resources provided in the environment are interesting for this perspective. Only their subset that has a direct influence on the agent autonomy should be included and the others should be hidden in order to limit the representation complexity.

It can be useful to remark that while the listed elements play a relevant role in characterizing the agent autonomy some others could be neglected; this is the case of agents communications (with the underlying transport mechanism, content language and interaction protocols), mechanism of knowledge updating, implementation architectures, and so on.

In order to clarify our concepts, let us abstract the execution of an autonomous agent as a movement in some abstract '*actions space*' (Fig. 1). The

mission of the agent (i.e., its vocation), for which it exploits all necessary knowledge, ability, and resources, is to move step by step in the action space (i.e., via a sequence of autonomous actions) until it reaches the goals positioned in this space. Clearly, depending on the actual decision-making of the agents, the goals can be reached by following various paths (i.e., via different combinations of its possible actions).

A concrete example of this generic agent and its actions space could be a robotic agent that is devoted to the exploration of some environment in order to collect information about its topology (walls, doors) and the position of furniture elements in it. It could move around following different paths and, using its sensors (laser, infra-red, sonar or even vision), it can discover the presence of different objects that it will classify according to its (a-priori) knowledge. In this simple example, the very goal of the agent it to fully explore the building, and autonomy of the agent is mainly used to let it decide in which way to explore the building, room after room. In other words, in this example, the abstract trajectory the agent has to follow to reach its goal (as from Fig. 1) equates to a physical trajectory in space.

Thus, from the autonomy perspective, what is of interest is not the activity of collecting data and knowledge about objects in the environment, but mostly the activity of moving in such environment.

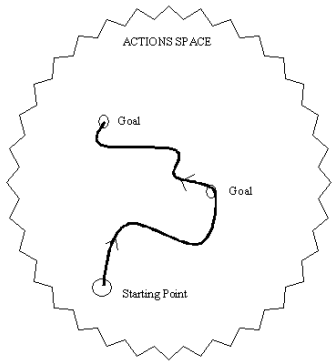

Fig. 1. An autonomous agent in an abstract trajectory towards its goal

3.3 Context

Each system is designed to solve one or more problems and this situates it in the context (usually referred as the problem domain) where those problems take place. Such a characterization particularly applies to agent, which have the peculiar characteristics of being entities situated in an environment, that is, of having an explicit representation of the context and of acting in it.

A common expedient used by designers to represent the context and the system interaction is the description of some operating scenarios. This could bring an enormous number of elements to the attention but only the part of them really affecting the specific perspective should be considered and the remaining other should be neglected.

The context in which agents of a MAS situates (whether a virtual computational environment like an e-commerce marketplace or a physical one like a building to be explored) introduces in the system design some constraints. These could be rules of the environment itself (e.g., an agent should pay for the good he won in an auction) or possible environment configurations that could effect the agent activities (e.g., fog could limit vision of a robotic agent). Moreover, the data an agent can acquire from the environment can be of some relevance too.

Since autonomous agents could be not deterministic, and since the agent decisional process is something that could not always be easily deducted from a black-box external observation of the agent behavior, a perspective on resources centered around the autonomy concept should take into account this aspect. In particular, when the focus is on the autonomous actions of the agents, the characteristics of the environment (and of the data that can be found there) of interest are those that can somewhat influences the agents decisional process. In other words, by taking into account the fact that the agent executes in an environment, that may influence it and may be influenced by it, an autonomy perspective would prescribe to identify what in the environment and its data could comes to interplay with the dimension of autonomy of the agent.

Going back to our explorer robotic agent, the topological and physical characteristics of the building) to be explored could affect the activities of the agent. A very large open site may enable an agent to explore in detail all possible objects in it. A site with objects in not accessible position will prevent him to do his work in a complete satisfactory way.

However, from the autonomy perspective, the data that are really of interest are the topological information about the environment, because these will influence the way the robot will find its way through the building. For instance, the presence of a ground slope that the agent cannot safely walk through may require him to take specific exploration choices or, which it the same, to modify the physical (and abstract) trajectory of its autonomous decisions. Other information like the color of walls or the style of furniture are not generally relevant for this perspective. Sometimes, it may be the case that the system requirements calls for a *curious* robot, capable of deciding to explore some specific objects more in detail using all of its sensing capabilities and possibly requiring it to step back to analyze relations with already analyzed objects. In such a case, further characteristics other than the topology of the environment may come into play influencing the agent autonomous actions.

3.4 Rationale

We already discussed that, during his activity, the designer aims at reaching some goals for the system; all of his choices will be guided by a precise strategy (one of the many possible ones) that he considers the best solution to the problem. In the context of an architectural perspective (looking at the best architectural solution) of a system devoted to provide the control of some active network routers this mean using many small well specialized agents that will not overload the network traffic rather than multiple instances of the same big all-purpose agent. In the case of an autonomy perspective, the rationale that is behind all the design activity is the decisional process used by agents (and imposed to them by the designer) to reach their goals. In the context of cooperative agents like the ones used in Adelfe [2], this means looking for a cooperative solutions while in other approaches and agent could prefer to face the problems by itself. At the end the decisional process will decide in which way the perspective elements (tasks, roles, ...) will be composed to satisfy the agent's vocation (another element of this perspective)

Considering the robot example, the decisional process is first of all characterized by the chosen cognitive architecture that will select the plan (for example a subsumption machine) and the strategy imposed to it by the designer. Let us suppose that the robot is not exploring the environment in order to collect new data about its topology but it is looking for bags forgotten by public in an airport. The mission is almost the same (finding new objects in the environment) but the decisional process could be different. For example, a new line of chairs is not considered an interesting element while a bag left alone in a crowded place could be a potential danger and therefore it requires an immediate attention by the robot that could even warn security personnel of the discovery.

In this case, the rationale determines how the path to the goal (as described in Fig. 1) is to be formed, and how it can be influenced by external factors.

In other words, it determines the way an agent finds a path toward its goal, and may also determine the way an agent may not be allowed to find a path, because this clashes with some requirements (giving attention to a not dangerous new line of chairs slows the surveillance of the assigned area) or because this is not made possible by the structure of the available resources.

3.5 Constraints

In a MAS, several agents execute in the same environment, towards the achievement of individual goals that may either contribute to a global application goal or that may be selfish goals. Whatever the case, the autonomy of agents does not imply that agents can do whatever they want independently of the actions of other agents. Rather, since agents live in the same universe, the 'trajectories' they follow should be disciplinated and not 'clash" with each other. In other words, in a MAS, the autonomy of agents should be somewhat reduced or 'adjusted' in order for the whole system to proceed correctly, by disciplining the

abstract trajectories that each agent would follow toward the achievement of the task.

A typical example of this is in the concept of 'social laws' introduced by Moshe and Tennenholtz [16, 13], which perfectly suit our example of mobile robots exploring an environment. There, each robot in a group of mobile robot – each having the selfish goal of exploring an environment – is disciplined in its movements (that is, in its autonomy) via the superimposition of social laws (traffic laws in the specific example) that prevent it for planning motion actions that would somewhat disturb the movements of other robots.

Another example is the concept of 'organizational rules' introduced in the latest version of the Gaia methodology [19]. There, in the analysis phase of a MAS, the modeling of the internal activity of each agent in a MAS (including the goals and tasks of each agent) has to be coupled with an explicitly modeling of the external rules that the system as a whole has to ensure. Clearly, such 'organizational rules' have to be somewhat enacted in the subsequent agent design by limiting the autonomy of those agent that would otherwise be at risk of breaking the organizational rules.

4 Conclusions and Future Works

The number of different issues that a designer is forced to face in the development of a MAS may require adopting a multi-perspective approach to system design. In particular, among a number of perspectives that can be conceived, a specific perspective focusing the issue of agent autonomy may be required to effectively tackle the peculiar characteristics of agents and of their being autonomous entities interacting in a complex world.

Having sketched the key characteristics of an autonomy perspective for MAS system design, as we have done in this paper, is only a first step. Further work will be required to make such a perspective applicable to a variety of current agent-oriented methodologies, such as PASSI [6], GAIA [17, 19], TROPOS [4]or MASE [7]. In addition, it will be important to verify on real-world applications the extent of applicability of the autonomy perspective and its possible limitations.

References

1. L. Bass, P. Clements, and R. Kazman. *Software Architectures in Practice (2nd Edition)*. Addison Wesley, Reading (MA), 2003.
2. C. Bernon, M.P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe, a methodology for adaptive multi-agent systems engineering. In *Proceedings of the Third International Workshop Engineering Societies in the Agents World (ESAW-2002)*, Madrid, Spain, September 2002.
3. G. Booch. *Object-oriented Analysis and Design (second edition)*. Addison Wesley, Reading (MA), 1994.
4. J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: The tropos project. In *To appear in Information Systems*, Elsevier, Amsterdam, The Netherlands, 2002.

5. M. Cossentino. Different perspectives in designing multi-agent systems. In LNCS, editor, *Proceedings of the AGES '02 workshop at NODe02*, Erfurt, Germany, October 2002.
6. M. Cossentino and C. Potts. A case tool supported methodology for the design of multi-agent systems. Las Vegas (NV), USA, June 24-27 2002. The 2002 International Conference on Software Engineering Research and Practice, SERP'02.
7. S. A. DeLoach, M. F. Wood, and C. H. Sparkman. Multiagent systems engineering. *International Journal on Software Engineering and Knowledge Engineering*, 11(3):231–258.
8. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns.* Addison Wesley, 1995.
9. C. Iglesias, M. Garijo, J. C. Gonzales, and J.R. Velasco. Analysis and design of multi-agent systems using mas-commonkads. In *Intelligent Agents IV: Agent Theories, Architectures, and Languages*, volume 1365, pages 313–326. Springer Verlag, 1998.
10. N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, April 2001.
11. N.R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117, 2000.
12. J. Kephart. Software agents and the route to the information economy. *Proceedings of the National Academy of Science*, 99(3):7207–7213, May 2002.
13. Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(3):533–562, 1995.
14. A. Newell. The knowledge level. *Artificial Intelligence*, 18, 1982.
15. H.V.D. Parunak. Go to the ant: Engineering principles from natural agent systems. *Annals of Operations Research*, 75:69–101, 1997.
16. Y. Shoham and M. Tennenholtz. Social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73, 1995.
17. M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–315, 2000.
18. M.J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
19. F. Zambonelli, N. R. Jennings, and M.J. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):417–470, July 2003.
20. F. Zambonelli, N.R. Jennings, and M.J. Wooldridge. Organizational abstractions for the analysis and design of multi-agent systems. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, volume 1957 of *LNCS*, pages 253–252. Springer Verlag, 2001.