University of Liege
Electrical Engineering and Computer Science Department
Sart-Tilman B-28
4000 Liege, BELGIUM

Agents and Multi-Agent Systems: A Short Introduction for Power Engineers

-Technical Report-

Dr. Mevludin Glavic

May, 2006

CONTENTS:

1. Introduction 3

2. What is an agent? 3

3. What is a Multi-Agent System (MAS) 7

4. What are advantages of MAS? 8

5. Independent discrete MAS 8

6. Independent MAS with emergent cooperation 8

7. Cooperative MAS 10

8. Homogeneous non-communicating MAS 10

9. Homogeneous communicating MAS 12

10. Heterogeneous non-communicating MAS 13

11. Heterogeneous communicating MAS 13

12. Software agents 14

13. Agent communication language 16

14. Architectures for software MAS 17

15. Other organizational paradigms 18

16. Considerations of agents and multi-agent systems in power system engineering 20

17. References and useful links 20

1. Introduction

More than twenty years of research in the field of agents and multi-agent systems have offered remarkable results from both theoretical and practical point of view. However, after so many years of the research there is no general consensus on some basic notions such as "what is an agent?", "what is a multi-agent systems", agreement on the terminology used, etc. Indeed, this fact makes confused those interested in applying agent based or multi-agent based technology to solve practical problems. This short note is intended to serve as a "gentle" introduction to the field of agents and multi-agent systems particularly for those interested in using these technologies in solving practical engineering problems. The main purpose of the note is to familiarize readers with basic terminology and definitions. This note will not enumerate (many) and elaborate different views on the subject but rather presents a view on the subject that (according to authors of the note) may help engineering people to get feeling on this important subject. Throughout of this note, when deemed necessary, simple examples are provided to illustrate different concepts. All the examples are based on the idea to use agent or multi-agent technology to design effective load shedding scheme. For readers interested for a deeper inside in the subject, from different points of view, this note offers at the end a list of useful references and links.

2. What is an agent?

There is no strict definition about what an agent is. Literature offers a variety of definitions ranging from the simple to the lengthy and demanding ones. All available definitions are strongly biased by the background field interested in agent technology (main being: artificial intelligence, software engineering, cognitive science, computer science, engineering in general, etc.).
Instead to list and elaborate different definitions, two definitions of the agent coming from [1] and [2] are given here since they seem to be rather general, widely accepted by different research communities, and closest to power engineering people perception of agents. In [1], an agent is defined as follows,

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors".

According to this definition an agent is any entity (**physical** or **virtual** one) that senses its environment and acting over it.
Physical entities that could be considered as agents are, in the case of a power system, simple protection relay or any controller that controls directly particular power system component or part of the system.
Virtual entity that can be considered as an agent is a piece of software that receives inputs from an environment and produces outputs that initiate acting over it.
Often an agent is a combination of physical (computation architecture) and virtual one (a piece of software running on the computational architecture).
A more precise definition of the agent, that is in fact extension of the definition mentioned above, is given in [2],

"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are deigned".

The key extensions in this definition with respect to first one are words: **computational**, **autonomy**, and **goals**. Word **computational** makes difference in agents that we are interested in engineering (**computational agents**) from biological agents (humans, animals, bacteria) since from first definition this distinction is not obvious. **Autonomy** means that computational agents operate without the direct intervention of some other entities and have some kind of control over their actions. Assigning the **goals** to the agent means that acting upon environment should be done in order to achieve some specified objective (goal) and that the agents expose a sort of **rational** (an agent that minimizes or maximizes its performance measure) behavior in the environment. **Behavior** here means the action that is performed after receiving sensory inputs (or any sequence of sensory inputs) [1]. A general single-agent framework is illustrated in Figure 1.

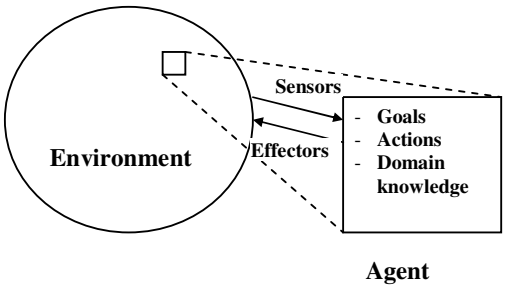


Figure 1. A general single-agent framework.

In addition to the sensory inputs, actions, and goals an agent may include domain knowledge (the knowledge about particular environment or problem to be solved). This knowledge can be algorithmic, artificial intelligence (AI) technique based (rule-based, fuzzy, neural networks, machine learning), heuristics, etc. In AI case an agent is often termed as **intelligent** one.

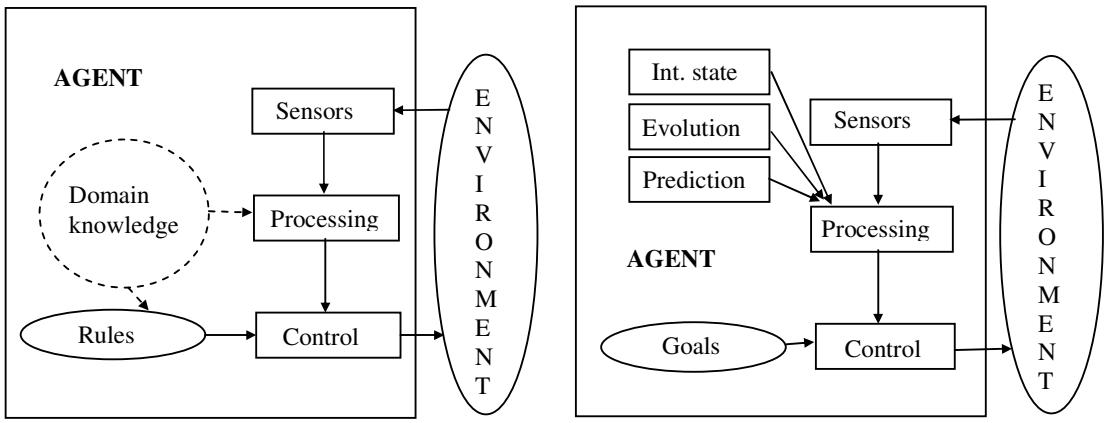
The notion of environment that an agent inhabits (is situated in or simply said placed in) include physical systems (as in engineering), operating system, the Internet, or perhaps some of these systems combined.

Observe also that physical entities and virtual ones can be combined to constitute an agent.

If an agent simply responds, in timely fashion, to changes in the environment and reactively converts its sensory inputs to actions, this agent is known as **reactive** (sometimes termed **reflex** agent). Reactive agents usually do not maintain internal state (a simple example of internal state is keeping a set of previous sensory inputs) of the agent and do not predict the effect of the actions. For can be said is that if an agent predicts the effect of the actions then it is not reactive (meaning that reactive agent may also maintain its internal state).

On the other hand, if an agent maintains internal state predicts the effects of its actions, or more generally includes a sort of reasoning (behave more like they are “thinking”), that agent behaves deliberatively and is termed as **deliberative** agent (or agent that keeps the track of environment).

Figure 2 illustrates reactive and deliberative agents. Some of the authors do not assign any goals to reactive agents but the goals can be embodied in the sensory inputs processing or condition-action rules.



a) Reactive

b) Deliberative

Figure 2. Reactive (reflex) and deliberative agents

The various definitions of agents involve a host of properties of an agent and agents are usually classified based on those properties.
Table I lists several properties that can be encountered in literature and are given here for the reader just to have an idea what the particular term (often considered as a type of agent) means.

Property	Other names	Meaning
Reactive	Reflex, sensing and acting	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions
Goal-oriented	Pro-active, purposeful	Does not simply act in response to the environment
Temporally continuous		Is a continuously running process
Communicative	Socially able	Communicates with other agents
Learning	Adaptive	Changes its behavior based on its previous experience
Mobile		Able to transport itself from one machine to another (this is associated manly with software agents)
Flexible		Actions are not scripted

Table I. Some properties of agents

Example 1

Suppose we want to design a simple load shedding agent (see figure 3). First we have to decide what sensory inputs and controls are going to be, what algorithm or rule to use to map sensory inputs to controls (domain knowledge). Let the control be all available load at considered bus that can be shed at once, and let the sensory input be the voltage magnitude at the bus where considered load is connected.

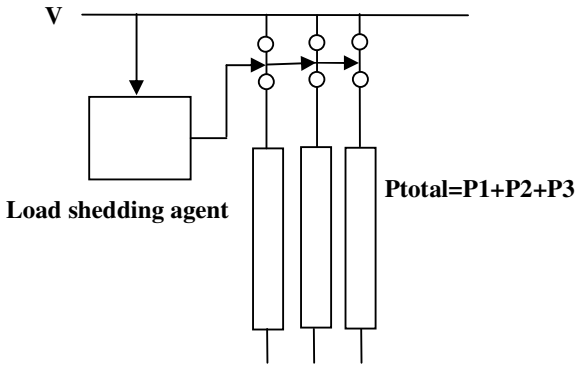


Figure 3. A simple load shedding agent

Let define a simple rule (domain knowledge):

IF $V \le V^{\min}$ THEN Shed P_{total}

This is typical reactive agent with no internal state (just reacting to the current value of the input). This agent could be realized using simple undervoltage relay (in this case we talk about physical agent) or by using a sort of computational architecture on which a very simple peace of code is implemented (in this case we talk about an agent that is combination of physical and virtual entities). With respect to other agent attributes given in Table I this agent is: not flexible (the control is scripted, i.e. shed all available load at once), is autonomous, not adaptive, and is temporally continuous since it permanently senses the voltage magnitude.

Example 2

Let now change the rule for load shedding and define it as,

IF $V \le V^{\min}$ THEN IF $P = k \cdot (V^{\min} - V) \le P_{total} - P_{already\_shed}$ Shed $P = k \cdot (V^{\min} - V)$
ELSE Shed Remaining load.

With respect to example 1 now the agent is flexible since the amount to be shed is not scripted and depends on the voltage magnitude. Also in this case the agent should maintain internal state in terms of the load amount already shed. Observe that in this case agent sheds enough load to keep the voltage magnitude above $V^{\min}$ so it has a goal (objective). All other attributes keep the same as in example 1. Further extensions are possible. For example we might decide that agent tracks the voltage magnitude changes and base its decisions not only on the voltage magnitude but also on the rate of magnitude change (this is just a possibility that is probably not justified in the case of undervoltage load shedding and serves just as an illustration). In this case the internal state of agents is extended to include certain number of past sensory inputs (moving window). If such an agent is able to predict the effect of actions it becomes deliberative agent (for example if voltage magnitude is below the threshold but voltage magnitude is increasing the agent might decide not to shed and wait for some time hoping the voltage will really recover).

3. What is a Multi-agent system (MAS)?

As for the definition of an agent the same holds for definition of a multi-agent system. Various definitions have been proposed depending on research discipline it is coming from. Again, the aim here is not to list and elaborate different definitions and rather a single definition is chosen that seems to be general and close to power engineering research community. In [3], a multi-agent system is defined as,

"A multi-agent system is a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity (agent)".

The fact that the agents within a MAS work together implies that a sort of cooperation among individual agents is to be involved. However, the concept of cooperation in MAS is at best unclear and at worst highly inconsistent, so that the terminology, possible classifications, etc., are even more problematic then in the case of agents what makes any attempt to present MAS a hard problem.
A typology of cooperation from [4] seems the simplest and here we start with this typology as the basis for MAS classification. The typology is given in Figure 4.

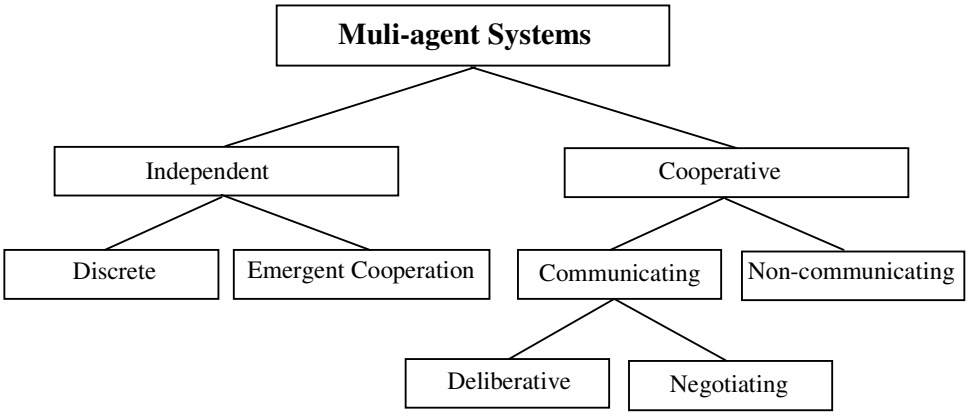


Figure 4. Cooperation typology

A MAS is independent if each individual agent pursues its own goals independently of the others. A MAS is discrete if it is independent, and if the goals of the agents bear no relation to one another. Discrete MAS involve no cooperation. However agents can cooperate with no intention of doing so and if this is the case then the cooperation is emergent. The term deliberative in Figure 4 is not to be mixed with the term of deliberative agent, and in the figure it refers to the idea that agents jointly plan their actions so as to cooperate with each other. The cooperation within a MAS can be realized in three ways:

- by explicit design (the designer of the agents purposely designs the agent behaviors so that cooperation occurs),
- by adaptation (individual agents learn to cooperate),
- by evolution (individual agents cooperation evolves through some kind of evolutionary process).

4. What are advantages of MAS?

Main advantages of MAS are robustness and scalability. Robustness refers to the ability that if control and responsibilities are sufficiently shared among agents within MAS, the system can tolerate failures of one or more agents.
Scalability of MAS originates from its modularity. It should be easier to add new agents to a MAS than to add new capabilities to a monolithic system.

5. Independent discrete MAS

This type of MAS is encountered in the environments that permit decoupling or decomposition (spatial, temporal). As an example of this type of MAS we use two agents (controllers) controlling synchronous generator in a power system: automatic voltage regulator (AVR) and speed governor. They have different goals that bear no relation to one another. AVR goal is to keep terminal voltage at predefined value and speed governor goal is to keep angular speed at synchronous value. AVR exercises control over the excitation system while speed governor exercises control by controlling inflow of primary medium (steam, water). They are designed independently and based on engineering observation of temporal decomposition (AVR acts much faster) and coupling between active power and speed as well reactive power and voltage magnitude). Of course one might argue that these agents are not completely independent but here serve to illustrate an example that looks like independent discrete and to point out prerequisite of decomposition and decoupling to realize this type of MAS.

6. Independent MAS with emergent cooperation

In this MAS agents are designed independently and each individual agent pursues its own goals independently of the others. It is more important to stress that in this MAS the individual agents are not aware of existence of other agents and each agent considers others as a part of the environment. Since agents exist in the same environment they can affect each other indirectly and the cooperation can emerge with no intention of doing so. The cooperation among independent agents can emerge in two ways:

- individual agents receive as sensory inputs (directly or they estimate them) the inputs or control of one or more other agents in the environment,
- individual agents, by their actions, change sensory inputs of another agent in a cooperative way without explicit intention of doing so.

Let us use again simple example to illustrate how cooperation can emerge in individual MAS.

Example 3

We now consider two load shedding agents placed at different buses (figure 5).

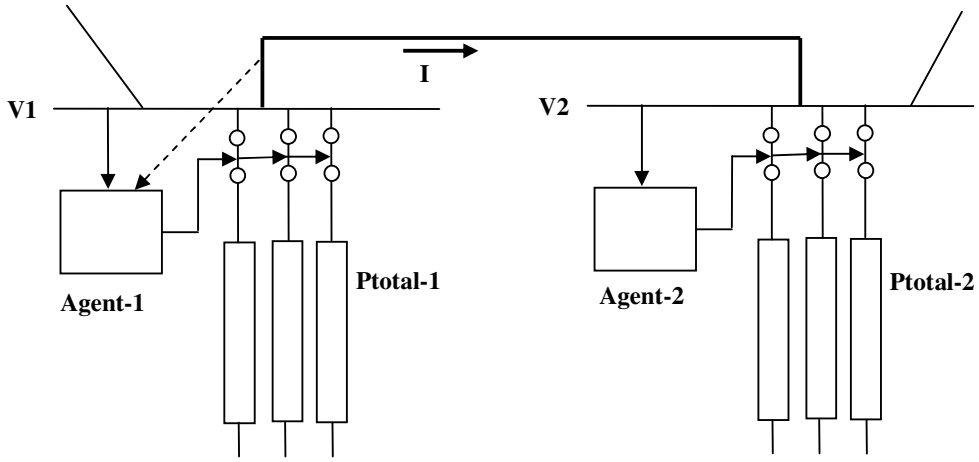


Figure 5. Two individual agents with emergent cooperation

One possibility for cooperation between these two agents to emerge is that agent-1 estimates voltage magnitude (sensory input of agent-2) by measuring current I ($V_2 \approx V_1 - I \cdot Z_{line}$) and bases its decision (control) not only on the magnitude of local voltage but also on the magnitude of V_2 (for example on average value of these two voltages). Agent-2 bases its decision only on local voltage magnitude (if both have same inputs in the form of average voltage they are in fact single agent, see latter in the text discussion on this issue). Since these agents have different sensory inputs, regardless of the fact that they may have some domain knowledge and actions they are **heterogeneous**. Suppose that decision rule for agent-1 is,

IF $V_{av} \le V_{av}^{\min}$ THEN IF $P = k \cdot (V_{av}^{\min} - V_{av}) \le P_{total}1 - P_{already\_shed}1$ Shed $P = k \cdot (V_{av}^{\min} - V_{av})$
ELSE Shed Remaining load.

and for agent-2,

IF $V_2 \le V_2^{\min}$ THEN Shed $P_{total}2$

If the voltage V_2 goes below specified threshold agent-2 will shed all the load in bus 2. If this shedding is enough to recover voltages agent-1 will not act, otherwise agent-1 will also shed some load if average voltage is below respective threshold, and helping the agent-2.
This is an example how cooperation is realized by explicit design (it is purposely designed that one agent estimates input of another).
Let now consider the case where both of the agents (figure 5) sense respective voltage magnitudes only, they have same actions (load shedding) and they have the same domain knowledge, or in other words they are **homogeneous** but differently situated (placed) in the system (environment). Since the agents have the same domain knowledge they use same decision rule, and let it be for agent-1,

IF $\int_0^t (V_1^{\min} - V_1)dt \ge C$ THEN IF $P = k \cdot (V_1^{\min} - V_1) \le P_{total}1 - P_{already\_shed}1$ Shed $P = k \cdot (V_1^{\min} - V_1)$
ELSE Shed Remaining load.

and for agent-2,

IF $\int_0^t (V_2^{\min} - V_2)dt \geq C$ THEN IF $P = k \cdot (V_2^{\min} - V_2) \leq P_{total}2 - P_{already\_shed}2$ Shed $P = k \cdot (V_2^{\min} - V_2)$

ELSE Shed Remaining load.

Cooperation in this case emerges from the fact that load shedding in either bus will improve voltage magnitude not only in local bus but also in its neighboring buses (we suppose here that the buses are electrically close), it affects sensory inputs of other agent in "positive manner" and at the end affects the action of other agent.
Time when individual agents will act is not scripted in this setting (otherwise if we design these times and to be different then we have intention that agents cooperate and cannot be considered as MAS with emergent cooperation, see section on homogeneous non-communicating MAS latter in the text). Suppose that the voltage decrease is more pronounced at bus 1 than in bus 2, the condition-action rule will be earlier satisfied for agent-1 and it will act, both voltage magnitudes will raise and hopefully voltage V_2 will go above respective threshold and agent-2 at best will not act at all or at worst will act with minimum shedding. Putting the same C for both agents facilitates emergence of cooperation (if we choose to put different C it is likely we put intention on cooperation by design and cooperation is intentional, not emergent).

7. Cooperative MAS

Since the aim of this note is to present the subject as clearly as possible to be understandable for power system researchers, the choice in the remaining of this text is to stick with presenting cooperative MAS from two dimensions: agent heterogeneity and amount of communication among individual agents within a MAS. With respect to these two dimensions cooperative MAS can be classified in four groups:

1. **Homogeneous non-communicating MAS,**
2. **Homogeneous communicating MAS,**
3. **Heterogeneous non-communicating MAS, and**
4. **Heterogeneous communicating MAS.**

8. Homogeneous non-communicating MAS

In this MAS there are several different agents with identical structure. All individual agents have the same goals, domain knowledge, and possible actions. They also have the same procedure for selecting among their actions. The only differences among individual agents are their sensory inputs and the actual actions they take, or in other words they are situated (placed) differently in the environment.

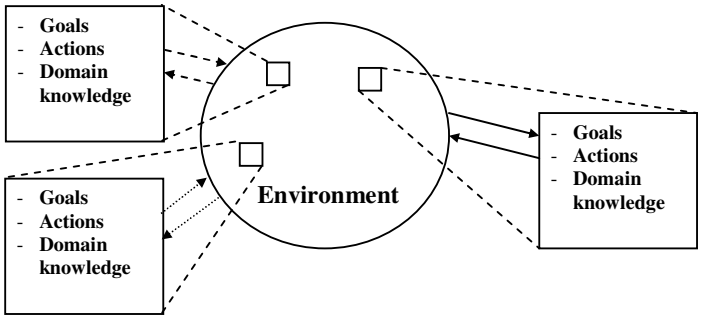


Figure 6. Homogeneous non-communicating MAS

A MAS with homogeneous non-communicating agents is illustrated in Figure 6 (the fact that individual agents have the same goals, actions, and domain knowledge, is indicated in the figure by representing them with identical fonts).
Individual agents in this MAS make their own decisions regarding which actions to take. Having different actions (outputs) is a necessary condition for MAS since if individual agents all act as a unit, then they are essentially a single agent.
In order to realize this difference in output, homogeneous agents must have different sensory inputs as well. Otherwise they will act identically.

Example 4

As a very simple example to illustrate homogeneous non-communicating MAS we first use very simple example of two overcurrent relays placed at the same feeder (figure 7).

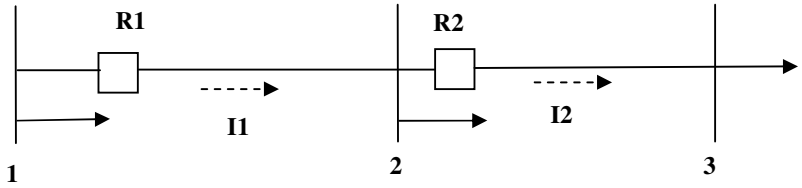


Figure7. Simple homogeneous non-communicating MAS.

Both relays are physical agents acting according to the rule,

IF $I_f \geq I_p$ TRIP the Breaker

where I_f is current flowing though the relay and I_p predefined threshold. To allow these agents to selectively trip only faulted part of the feeder different time delays for the trip are assigned to the relays and one more rule is added to the relay R1. So the rules are,

IF $I_1^1 \geq I_{p1}$ TRIP the Breaker after delay t_1 AND
IF $I_1^2 \geq I_{p2}$ TRIP the Breaker after delay t_2

IF $I_2 \geq I_{p2}$ TRIP the Breaker after delay t_3

Time delays t_1 and t_3 are small with respect to t_2 so the relays could timely trip the breaker if a fault occurs near the relay but if relay R2 fails relay R1 will trip after longer delay.
So, in this case individual agents are aware of the existence of other agents (and they are not independent), they do not communicate but cooperate since in the design phase they are aware of each other, they are homogeneous. The main difference between homogeneous non-communicating MAS and independent MAS with emergent cooperation is the awareness of existence of other agents, for cooperating MAS, in the design phase and using its domain knowledge the designer make individual agents cooperate with intention of doing so.
Similarly we can design load shedding scheme by assigning different delays for load shedding to different agents. Let now consider the case where both of the agents (figure 5) sense respective voltage magnitudes only, they have same actions (load shedding) and they have the same domain knowledge. Since the agents have the same domain knowledge they use same decision rule, and let it be for agent-1,

IF $V_1 \le V_1^{\min}$ during time t_1 THEN IF $P = k \cdot (V_1^{\min} - V_1) \le P_{total}1 - P_{already\_shed}1$ Shed $P = k \cdot (V_1^{\min} - V_1)$
ELSE Shed Remaining load.

and for agent-2,

IF $V_2 \le V_2^{\min}$ during time t_2 THEN IF $P = k \cdot (V_2^{\min} - V_2) \le P_{total}2 - P_{already\_shed}2$ Shed $P = k \cdot (V_2^{\min} - V_2)$
ELSE Shed Remaining load.

Suppose that both voltages go below respective thresholds and let $t_1 < t_2$, agent-1 will act after time t_1, both voltage magnitudes will raise and hopefully voltage V_2 will go above respective threshold and agent-2 at best will not act at all or at worst will act with minimum shedding. Again the agents are homogeneous, do not communicate but intentional cooperation is realized by assigning different time delays for load shedding.

9. Homogeneous communicating MAS

In this MAS individual agents can communicate with each other directly. With the aid of communication, agents can coordinate more effectively. A MAS with homogeneous communicating agents is shown in Figure 8.

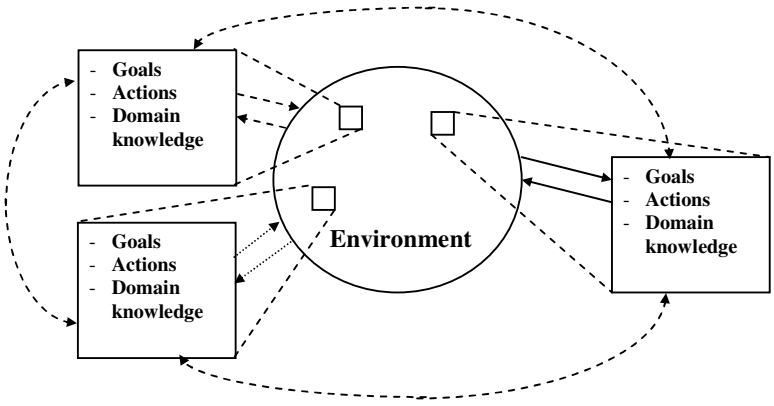


Figure 8. Homogeneous communicating MAS.

Communication raises several issues to be addressed in MAS. One of the most important issues is what the individual agents should communicate. Some possibilities are: communicate only sensing to each other, share information regarding individual goals, etc. Also it is possible for individual agents to learn what and when to communicate with other agents within a MAS based on observed effects of the performances of MAS.

10. Heterogeneous non-communicating MAS

Individual agents within a MAS might be heterogeneous in a number of ways, from having different goals to having different domain knowledge and actions. Adding the possibility of heterogeneous agents in a MAS adds a great deal of potential power at the price of added complexity.
A MAS with heterogeneous agents is illustrated in Figure 9 (the heterogeneity is indicated by different fonts for goals, actions, or domain knowledge).

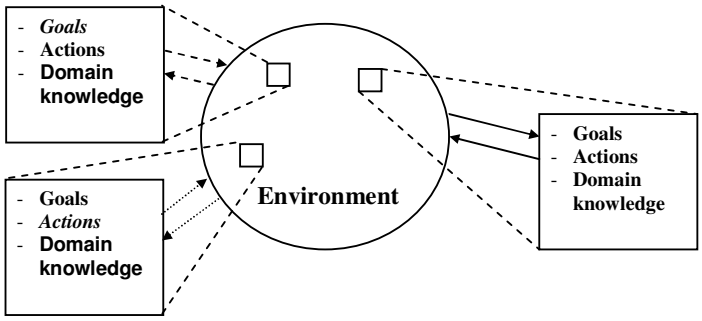


Figure 9. The heterogeneous non-communicating MAS.

Here, the individual agents are also situated differently in the environment which causes them to have different sensory inputs and necessitates their taking different actions. However, in this case the agents have much more significant differences. They may have different goals, actions, and/or domain knowledge.
One of the most important issues to consider when designing a MAS of this type is whether the different agents will be **benevolent** (willing to cooperate) or **competitive**.
Even if the individual agents have different goals, the agents can be benevolent if they are willing to help each other achieve their respective goals. On the other hands, the agents may be selfish and only consider their own goals when acting (competitive).

11. Heterogeneous communicating MAS

This type of MAS is shown in Figure 10. Individual agents (with different goals, actions, and /or domain knowledge) can communicate with one another. This MAS inherits the issues of communicating from homogeneous communicating MAS but heterogeneity brings additional issues to the communication. Two most important issues are: communication protocols and theories of commitment. Also, the issue of benevolence vs. competitiveness becomes more complicated for this type of MAS.

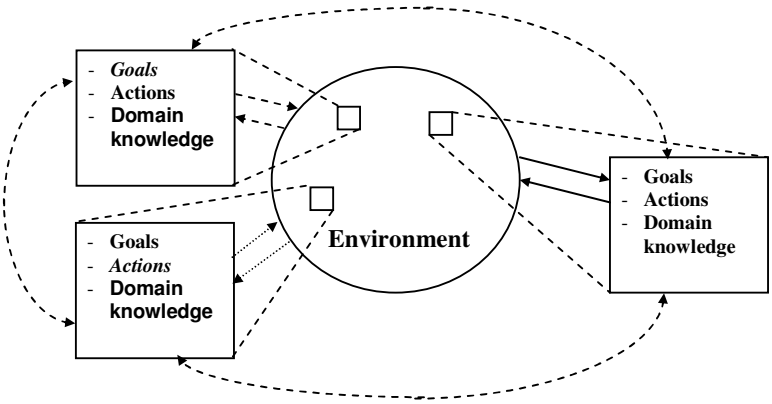


Figure 10. The general heterogeneous communicating MAS.

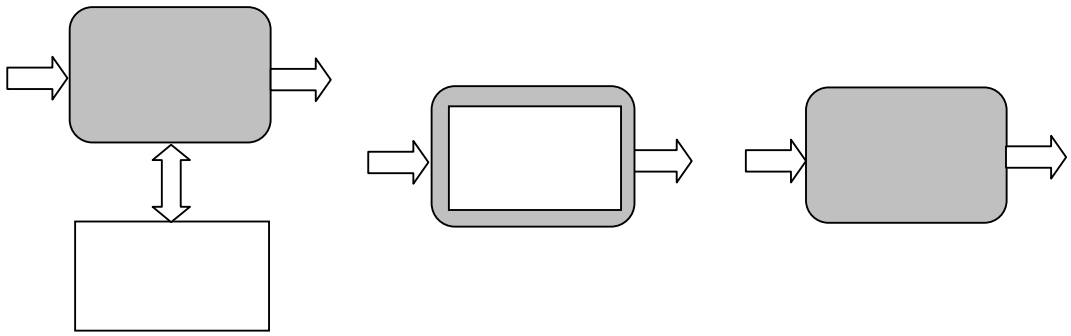
In all communicating MAS, and particularly in the case of heterogeneous individual agents, there must be some set language and protocol for individual agents to use when communicating.
Independent aspects of protocols are information content, message format, and coordination conventions. Most popular (and widely accepted) existing protocols for these three levels are: KIF (acronym for Knowledge Interchange Format) for content, KQML (acronym for Knowledge Query and Manipulation Language), for message format, and COOL (acronym for COOrdination Language) for coordination.
When agents communicate, they may decide to cooperate on a given task or for a given amount of time. In so doing, they make commitments to each other. Committing to another agent involves agreeing to pursue a given goal, possible in a given manner, regardless of how much it serves individual agent's own interests. There are, in general, three types of commitment: internal commitment – an agent binds itself to do something; social commitment – an agent commits to another agent (term **social agent** is often used); and collective commitment – an agent agrees to fill a certain role (term **team agent** is often in use).

12. Software agents

Software agents can be as simple as subroutines, but typically they are larger entities with some sort of persistent control. Usually, in software engineering, what makes agent-based software different with respect to "ordinary" software is its ability to interoperate and software agents are virtual (software) entities that communicate with their peers by exchanging messages in an expressive agent communication language. Agent-based software engineering was invented to facilitate the creation of this type of software.
This paradigm is often compared to object-oriented programming. Like an "object", an agent provides a message-based interface independent of its internal data structures and algorithms. The primary difference between the two approaches lies in the language of the interface. In general object-oriented programming, the meaning of a message can vary from one object to another. In agent-based software engineering, agents use a common language with an agent-independent semantics.
The concept of agent-based software raises a number of questions and most important ones are,

1. What is an appropriate agent communication language?
2. What communication "architectures" are conductive to cooperation?

Before addressing above questions let us first address very important issue, particularly from power engineering (but also other engineering disciplines) standpoint and that is the issue of existing (legacy) software. There is a vast panel of developed software capable to help solve many engineering problems. These tools are developed using procedural or object-oriented paradigms, they implement best algorithms and were (or still are) programmed using best programming skills in these paradigms. Their validity and usefulness is confirmed through many years of their use in practice. No one can discard any software tool just because it is not programmed using agent-oriented paradigm, and legacy software is to be agentified if one is interested in using them as a part of a MAS. Three possible approaches to agentification of legacy software are illustrated in Figure 11.



a) Converter (Transducer) b) Wrapper c) Rewrite

Figure 11. Three approaches to agentification

One approach (Figure 11a) is to implement a converter (sometimes termed transducer) that mediates between an existing program and other agents within a MAS. The converter accepts messages from other agents, translates them into the program's "understandable" variables or values, and passes those to the program, and of course vise versa, it accepts the program's response, translates into agent communication language and sends the resulting messages to other agents. This approach has the advantage that it requires no knowledge of the program itself, only its communication behavior (inputs and outputs). It is, therefore, especially useful for the situations in which the code for the program is unavailable or too delicate to modify.

A second approach to dealing with legacy software (Figure 11b) is to implement a wrapper, i.e. inject a new piece of the code into a program to allow it to communicate in agent communication language. The wrapper can directly examine the data structures of the program and can modify those data structures. Furthermore, it may be possible to inject calls out of the program so that it can take advantage of externally available information and services. This approach has the advantage of greater efficiency than the converter approach since there is less serial communication. It also works for cases where there is no interprocess communication ability in the original program. However, it requires that the code for the program be available.

The third approach (Figure 11c) is a drastic one and requires the original program to be rewritten in order to comply with new requirements. From engineering standpoint this is probably last resort and will not be further elaborated in this note.

13. Agent communication language

Communication language standards facilitate the creation of interoperable software by decoupling implementation from interface. As long as programs are committed to the details of the standards, it does not matter how they are implemented. Today, standards exist for a wide variety of domains. For example, electronic mail programs from different vendors manage to interoperate through the use of mail standards like SMTP.

Agent communication language (as any other standard) should assure consistency (the same words or expressions must have same meaning for all programs) and compatibility (all programs must use same words and expressions to "say" the same things).

There are two popular approaches to the design of agent communication language: the procedural approach and the declarative approach.

The procedural approach is based on the idea that communication can be best modeled as the exchange of procedural directives. Scripting languages (such as Perl and TCL) are based on this approach. They are both simple and powerful. They allow programs to transmit not only individual commands but entire programs. They are also (usually) directly and efficiently executable. Unfortunately, there are disadvantages to purely procedural languages and they are: sometimes the procedures require information about the recipient that may not be available to the sender, procedures are unidirectional (for agents the information share usually should be usable in both directions), scripts are difficult to merge.

The declarative approach to language design is based on the idea that communication can be best modeled as the exchange of declarative statements (definitions, assumptions, etc). To be maximally useful, a declarative language must be sufficiently expressive to communicate information of widely varying sorts (including procedures). At the same time, the language must be reasonably compact, it must ensure that communication is possible without excessive growth over specialized languages. In agent-based systems the use of declarative communication languages is paradigm to prevail but in short-term procedural approach will still be in rather wide use.

A declarative agent communication language 9sometimes called ACL) can best be thought of as consisting of three parts: its vocabulary (vocabulary implements what is known in agent literature as ontology, but the reader should be careful since one vocabulary can implement more than one ontology), an "inner" language called KIF, and an "outer" language called KQML. An agent communication language message is a KQML expression in which the "arguments" are terms or sentences in KIF formed from words in the vocabulary.

The vocabulary is listed in a large and open-ended dictionary of words appropriate to common application areas []. Each word in the dictionary has an English description for use by humans in understanding the meaning of the word, and each word has formal annotations (written in KIF) fro use by programs.

Note that the existence of such a dictionary does not imply that there is only one way of describing an application area. Indeed, the dictionary can contain multiple ontologies for any given area. An example of multiple ontologies is description three-dimensional geometry in terms of polar coordinates, rectangular coordinates, cylindrical coordinates, etc.

14. Architectures for software MAS

In a software MAS one of the issues is how individual agents should be organized to enhance cooperation. Two very different approaches have been explored: direct communication, in which agents handle their own coordination) and assisted coordination, in which agents rely on special system programs to achieve coordination.

The advantage of direct communication is that it does not rely on the existence, capabilities, or biases of any other programs. Two popular architectures for direct communication are the contract-net approach and specification sharing.

In the contract-net approach (also termed market-based organization), agents in need of service distribute requests for proposals to other agents. The recipients of these messages evaluate those requests and submit bids to the originating agents. The originators use these bids to decide which agents to task and then award contract to those agents.

In the specification sharing approach, agents supply other agents with information about their capabilities and needs, and these agents can then use this information to coordinate their activities. The specification sharing approach is often more efficient than contract-net approach because it decreases the amount of communication that must take place.

One disadvantage of direct communication is cost. As long as the number of agents is small, this is not a problem. But, in a setting with huge number of programs the cost of broadcasting the bids or specifications and consequential processing of those messages is prohibitive. In this case, the only alternative is to organize the agents in some way that avoids such broadcast. Another disadvantage is implementation complexity. In the direct communication schemes, each agent is responsible for negotiating with other agents and must contain all of the code necessary to support this negotiation. If only these capabilities could be provided by the system, this would lessen the complexity of application programs.

A popular alternative to direct communication that eliminates both above mentioned disadvantages is to organize agents into what is often called a federated system. A simple structure of such a system is illustrated in Figure 12. As suggested by the diagram, agents do not communicate directly but they communicate only with system programs called facilitators (or sometimes termed mediator, but it could be claimed that the facilitator is generalization of the concept of mediator).

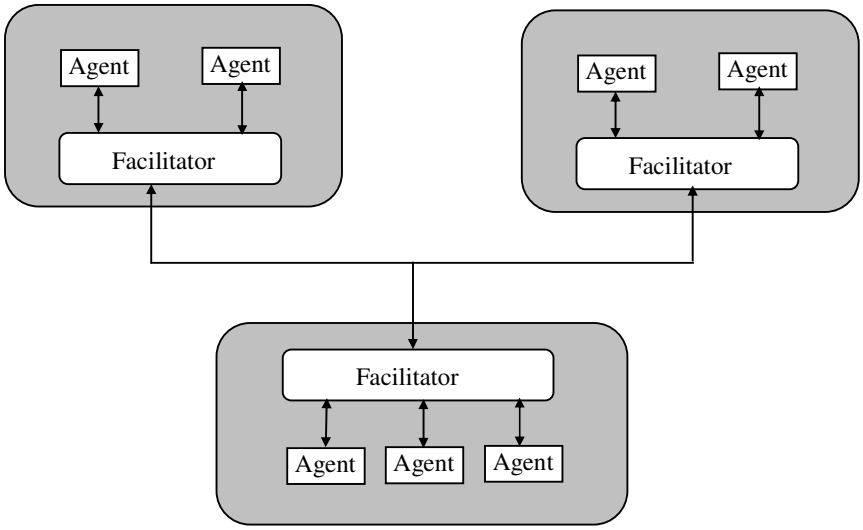


Figure 12. Federated system

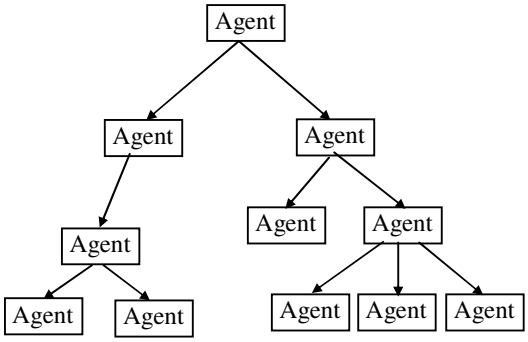
In this system, agents use agent communication language (in practice, a restricted set of this language) to document their needs and abilities for their local facilitators. In addition to this they also send application-level information and request to their facilitators and accept application-level information and requests in return. Facilitators use the documentation provided by these agents to transform these application-level messages and route them to the appropriate place. In effect, the agents form a "federation" in which they surrender their autonomy to their facilitators and the facilitators take the responsibility for fulfilling their needs.

15. Other organizational paradigms

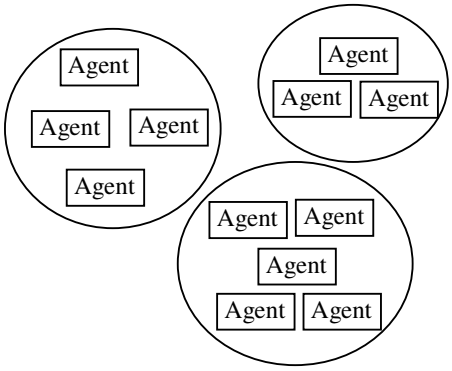
In addition to the organizational paradigms considered in previous section, several other paradigms emerged for MAS such as:

- Hierarchical organization,
- Coalitions,
- Teams,
- Congregations,
- Societies, and
- Compound organizations.

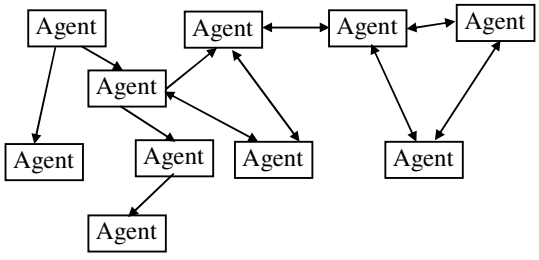
These agent organizations within a MAS are illustrated in Figure 13.



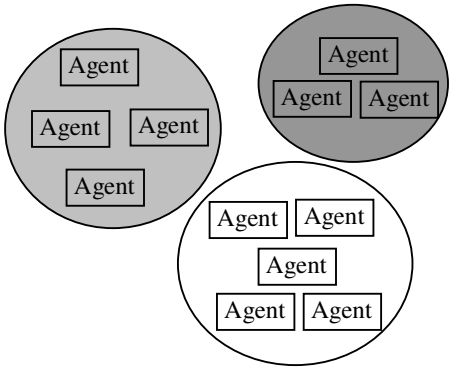
a) Hierarchical organization



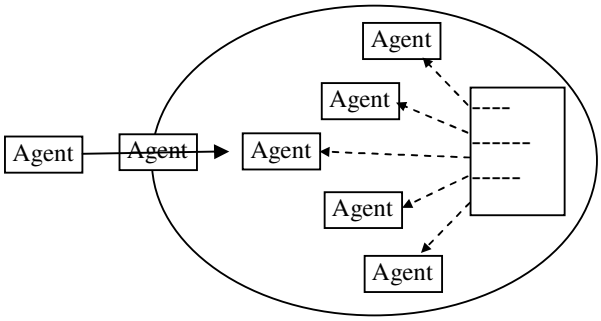
a) Coalitions



c) Team-based organization



d) Congregations



e) An agent society

Figure 13. Some other organizational paradigms of agents within a MAS

In hierarchical organization individual agents are organized in a tree-like structure, where agents higher in the tree have a more global view than those below them. Interactions do not take place across the tree. The data produced by lower-level agents in a hierarchy typically travels upwards to provide a broader view, while control flows downward.

In coalition type of organization the agents are grouped according to their goals into the groups of the agents having similar goals.

Within a coalition (Figure 13b), the organizational structure is typically flat, although there may be a distinguished "leading agent" which acts as a representative for the group. Once formed, coalitions may be treated as single entity. The agents in one group (coalition) are expected to coordinate their activities in a manner appropriate to the coalition's purpose. Coordination does not take place among agents in separate coalitions, except to the degree that their individual goals interact.

A team of agents (Figure 13c) consists of a number of cooperative agents which have "agreed" to work together toward a common goal. In comparison with coalitions, teams attempt to maximize the utility of the team itself, rather than that of the individual members. Agents are expected to coordinate in some fashion such that their individual actions are consistent with and supportive of the team's goal. Usually individual a=gents takes a role in achieving common goal.

Similar to coalitions and teams, agent congregations are groups of individual agents banded together into a typically flat organization in order to derive additional benefits. Congregations (Figure 13d) are formed among agents with similar or complementary characteristics. Different shadings in Figure 13d represent the potentially heterogeneous purpose behind each grouping, in comparison to typically homogeneous coalitions. Here, individual agents do not necessarily have a single or fixed goal. Analogous everyday examples of this type are: clubs, support groups, academic departments, etc.

Unlike other organizational paradigms, agent societies are inherently open systems. Agents have different goals and heterogeneous capabilities. In this paradigm there is a set of constraints that are imposed over the individual agents (Figure 13e), and these constraints are known as social laws, norms or conventions. The norms are in fact the set of rules or guidelines by which the agents must act, which provides a level of consistency of behaviors. For example the rules might constrain the type of protocols the agents use to communicate.

Compound organizational structures combine some of the above mentioned paradigms making the use of best characteristics of different organizational paradigms.

16. Considerations of agents and multi-agent systems in power system engineering

As with many other engineering fields, the power system research communities and practitioners are getting more and more interested in using these technologies to solve different kind of problems. This note will not elaborate individually all considerations and rather points out some publications where power system problems are tackled or solved using agent or multi-agent technologies. The reports on these considerations are included in references [5-13].
NOTE: All these organizational paradigms also hold for physical agents but are here discussed as the organizational paradigms of software agents.

17. References and useful links

1) S. Russel, P. Norvig, *"Artificial intelligence – A modern approach",* Prentice Hall, 1995.
2) P. Maess, *"Artificial life meets entertainment: Life like autonomous agents",* Communications of the ACM, vol. 38, no. 11, pp. 108-114, 1995.
3) P. Stone, M. Veloso, "Multiagent systems: A survey from a machine learning perspective", *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.
4) J. Doran, S. Franklin, N. R. Jenkins, T. J. Norman, "On cooperation in multi-agent systems", In *UK Workshop on Foundations of Multi-agent Systems*, Warwick, 1996.
5) C. Rehtanz (editor), *"Autonomous systems and intelligent agents in power system control and operation"*, Springer, 2003.
6) A. L. Dimeas, N. D. Hatziargyriou, "Operation of a Multiagent System for Microgrid Control", *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1447-1455, 2006.
7) H. F. Wang, H. Li, H. Chen, "Coordinated Secondary Voltage Control to Eliminate Voltage Violations in Power Systems Contingencies", IEEE Transactions on Power Systems, vol. 18, no. 2, pp. 588-595, 2003.
8) J. Jung, C. C. Liu, S. L. Tanimoto, V. Vittal, "Adaptation in Load Shedding under Vulnerable Operating Conditions", IEEE Transactions on Power Systems, vol. 17, no. 4, pp. 1199-1205, 2002.
9) H. Ni, G. T. Heydt, L. Mili, "Power System Stability Agents Using Robust Wide Area Control", IEEE Transactions on Power Systems, vol. 17, no. 4, pp. 1123-1131, 2002.
10) V. Krishna, V. C. Ramesh, "Intelligent Agents for Negotiations in Market Games, Part 1: Models", IEEE Transactions on Power Systems, vol. 13, no. 3, pp. 1103-1108, 1998.
11) S. Talukdar, V. C. Ramesh, "A multi-agent technique for contingency constrained optimal power flows", IEEE Transactions on Power Systems, vol. 9, no. 2, pp. 855-861, 1994.
12) C. C. Liu, J. Jung, G. T. Heydt, V. Vittal, A. G. Phadke, "The strategic power infrastructure defense (SPID) system: A conceptual design", IEEE Control Systems Magazine, vol. 20, no. 4, pp. 40-52, 2000.
13) H. Li, G. W. Rosenwald, J. Jung, C. C. Liu, "Strategic power infrastructure defense", Proceedings of the IEEE, vol. 93, no. 5, pp. 918-933, 2005.
14) K. P. Sycara, "Multiagent systems", *AI Magazine*, Summer 1998, pp. 79-92, 1998.
15) J. Ferber, *"Multiagent systems – An introduction to distributed intelligence",* Reading, MA, Addison-Wesley, 1999.
16) J. M. Bradshow*, "Software agents",* Cambridge, MA, MIT Press, 1997.
17) M. Wooldridge, *"An introduction to multiagent systems",* John Wiley and Sons, 2002.
18) M. R. Genesereth, S. P. Ketchpel, "Software agents", [Online] Available: www.cs.nott.ac.uk/~mhl/ archive/Genesereth+Ketchpel:94a.pdf
19) J. M. Bradshow, " An introduction to software agents", [Online] Available: www.cs.umbc.edu/**agents**/introduction/01-Bradshaw.pdf

20) Object Manager Group (OMG): www.objs.com/**agent**/
21) Foundation for Intelligent Physical Agents (FIPA): www.fipa.org
22) Knowledgeable Agent-oriented System (KAoS): citeseer.ist.psu.edu/bradshaw95**kaos**.html
23) General Magic group: **agent**s.umbc.edu/01/06/
24) www.aaai.org/AITopics/html/multi.html
25) www.**multiagent**.com/