Session 5

Towards Decentralization

Matti Saastamoinen
matti.saastamoinen@honeywell.com

The study of multiagent systems began in the field of distributed artificial intelligence (DAI)) about 20 years ago. Today these systems are not simply a research topic, but are also beginning to become an important subject of academic teaching and industrial and commercial application. DAI is the study, construction, and application of multi-agent systems, that is, systems in which several interacting,intelligence agents pursue some set of goal or perform some set of tasks [1].

This paper is based mostly on book Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence by Gerhard Weiss, Part I: Basic Themes.

5.1 Introduction

Artificial intelligence is an advanced form of computer science that aims to develop software capable of processing information on its own, without the need for human direction [2]. The term was first time used in 1956 by John McCarthy at the Massachusetts Institute of Technology.

Artificial intelligence includes:

- games playing: programming computers to play games such as chess and checkers

- expert systems: programming computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms)

- natural language: programming computers to understand natural human languages

- neural networks: Systems that simulate intelligence by attempting to reproduce the types of physical connections that occur in animal brains

- robotics: programming computers to see and hear and react to other sensory stimuli

An agent is a computational entity such as a software program or a robot that can be viewed as perceiving and acting upon its environment and is autonomous in that its behavior at least partially depends on its own experience. As an intelligent entity, an agent operates flexibly and rationally in a variety of environmental circumstances given its perceptual and effectual equipment. Behavioral flexibility and rationality are achieved by an agent on the basis of key processes such as problem solving, planning, decision making, and learning. As an interacting entity, an agent can be affected in its activities by other agents and perhaps by humans.

A key pattern of interaction in multiagent systems is goal- and task-oriented coordination, both in cooperative and in competitive situations. In a case of cooperation several agents try to combine their efforts to accomplish as a group what the individuals cannot, and in the case of competition several agents try to get what only some of them can have.

Two main reasons to deal with DAI can be identified, and these two reasons are the primary driving forces behind the growth of this field in the recent years. The first is that multiagent systems have the capacity to play a key role in current and future computer science and its applications. Modern computing platforms and information environments are distributed, large, and heterogeneous. Computers are no longer stand-alone, but have became tightly connected both with each others and their users. The increasing complexity of computer and information systems goes together with an increasing complexity of their applications. These often exceed the level of conventional, centralized computing because they require, for instance, the processing of huge amounts of data, or of data that arises at geographically distinct locations. To cope with such applications, computers have to act more as "individuals" or agents, rather than just "parts".

5.2 Intelligent agents that interact

"Agents" are autonomous, computational entities that can viewed as perceiving their environment through sensors and acting upon their environment through effectors. Agents pursue goals and carry out tasks in order to meet their design objectives, and in general these goals can be supplementary as well as conflict [3].

"Intelligent" indicates that the agents pursue their goals and execute their tasks such that they optimize some given performance measures. To say that agents are intelligent does not mean that they are omniscient or omnipotent, nor does it mean that they never fail. Rather, it means that they operate flexibly and rationally in a variety of environmental circumstances, given the information they have and their perceptual and effectual capabilities.

"Interacting" indicates that agents may be affected by other agents or perhaps by humans in pursuing their goals and executing their tasks. Interaction can take place indirectly through the environment in which they are embedded or directly through a sheared language. To coordinate their goals and tasks, agents have to explicitly take dependencies among their activities into consideration. Two basic, contrasting patterns of coordination are cooperation and competition. In the case of cooperation, several agents work together and draw on the board collection of their knowledge and capabilities to achieve a common goal. Against that, in the case of competition, several agents work against each other because their goals are conflicting. Cooperating agents try to accomplish as a team what the individuals cannot, and so fail or succeed together. Competitive agents try to maximize their own benefit at the expense of others, and so the success of one implies the failure of others. The following major characteristics of multiagent systems are identified:

- each agent has just incomplete information and is restricted in its capabilities;
- system control is distributed;
- data is decentralized; and
- computation is asynchronous.

Broadly construed, both DAI and the traditional AI deal with computational aspects of intelligence, but they do so from different points of view and under

different assumptions. Where traditional AI concentrates on agents as "intelligent stand-alone systems" and on intelligence as a property of systems that act in isolation, DAI concentrates on agents as "intelligent connected systems" and on intelligence as a property of systems that interact. In this way, DAI is not so much a specialization of traditional AI, but a generalization of it.

5.3 Rationales for multiagent systems

The two major reasons that cause to study multiagent systems are:

- *Technological and Application Needs* — Multiagent systems offer a promising and innovative way to understand, manage, and use distributed, large-scale, dynamic, open, and heterogeneous computing and information systems. The Internet is the most prominent example of such systems; other examples are multi-database systems and in-house information systems. These systems are too complex to be completely characterized and precisely described. DAI does not only aim at providing know-how for building sophisticated interactive systems from scratch, but also for interconnecting existing legacy systems such hat they coherently act as whole. Moreover, like no other discipline, DAI aims at providing solutions to inherently distributed and inherently complex applications [4].

- *Natural View of Intelligent Systems* — Multiagent systems offer a natural way to view and characterize intelligent systems. Intelligence and interaction are deeply and inevitably coupled, and multiagent systems reflect this insight. Natural intelligent systems, like humans, do not function in isolation. Instead, they are at the very least a part of the environment in which they and other intelligent systems operate. Humans interact in various ways and at various levels, and most of what humans have achieved is a result of interaction.

In addition, multiagent systems, as distributed systems, have the capacity to offer several desirable properties:

- *Speed-up and Efficiency* — Agents can operate asynchronously and in parallel, and this can result in an increased speed.

- *Robustness and Reliability* — The failure of one or several agents does not necessarily make the overall system useless, because other agents already available in the system may take over their part.
- *Scalability and Flexibility* — The system can be adopted to an increased problem size by adding new agents, and this does not necessarily affect the operationality of the other agents.
- *Costs* — It may be much cost-effective than centralized system, since it could be composed of simple subsystems of low unit cost.
- *Development and Reusability* — Individual agents can be developed separately by specialists, the overall system can be tested and maintained more easily, and it may be possible to reconfigure and reuse agents in different application scenarios.

The available computer and network technology forms a sound platform for realizing these systems. In particular, recent development in object-oriented programming, parallel and distributed computing, and mobile computing, as well as ongoing progress in programming and computing standardization efforts such as KSE, FIPA and CORBA are expected to further improve the possibilities of implementing and applying DAI techniques and methods.

5.4 Multiagent systems

5.4.1 Introduction

Agents operate and exist in some environment, which typically is both computational and physical. The environment might be open or closed, and it might or might not contain other agents. Although there are situations where an agent can operate usefully by itself, the increasing interconnection and networking of computers is making such situations rare, and in the usual state of affairs the agent interacts with other agents [5].

Communication protocols enable agents to exchange and understand messages. Interaction protocols enable agents to have conversations, which for our purposes are structured exchanges of messages. As a concrete example of these, a communication protocol might specify that the following types of messages can be exchanged between two agents:

- Propose a course of action

- Accept a course of action
- Reject a course of action
- Retract a course of action
- Disagree with a proposed course of action
- Counter propose a course of action.

5.4.2 Motivations

Why should we be interested in distributed systems of agents? Distributed computations are sometimes easier to understand and easier to develop, especially when the problem being solved is itself distributed. There are also times when a centralized approach is impossible, because the systems and the data belong to independent organization that want to keep their information private and secure for competitive reason [5].

The information involved is necessarily distributed and it resides in information systems that are large and complex in several sense: (1) they can be geographically distributed, (2) they can have many components, (3) they can have a huge content, both in the number of concepts and in the amount of data about each concept and (4) they can have a broad scope, i.e., coverage of a major portion of a significant domain. The topology of these systems is dynamic and their content is changing so rapidly that it is difficult for a user or an application program to obtain correct information, or for enterprise to maintain consistent information.

There are four major techniques for dealing with the size and complexity of these enterprise information systems: modularity, distribution, abstraction, and intelligence, i.e., being smarter about how you seek and modify information. The use of intelligence, distributed modules combines all four of these techniques, yielding a distributed artificial intelligence (DAI) approach.

For the practical reason that the systems are too large and dynamic for global solutions to be formulated and implemented, the agents need to execute autonomously and be developed independently.

5.5 Degree of decentralization

Here's a look at degrees of decentralization of agents that serve multiple, geographically-dispersed users [6].

5.5.1 A single central server

This is generally the easiest and most obvious organization for any sort of agent that serves multiple users.

Advantages:

- Easy to coordinate; no work has to be done by the agent itself to do so.
- Easy for users to know where to contact.
- Lends itself to crossbar algorithms and similar cases in which the entire knowledge base must be examined for each query or action.
- If the server is used by very widely spread users, timezones may spread out some of the load.

Disadvantages:

- Doesn't scale: generally, the workload goes up as the square of the number of users.
- Not fault tolerant: the server is a single point of failure for both performance and security (it is a single obvious point to compromise).
- The majority of users will find themselves a good part of an Internet diameter away from the server; this can be serious if low latency is required of the server.

5.5.2 Multiple mirrored servers

This describes a class of server where the basic algorithm is run in parallel on a number of machines (typically, very-loosely-coupled parallelism, e.g., separate workstations, rather than a single MIMD or SIMD parallel architecture). Such architectures in general can be divided into:

- Tightly-consistent architectures, in which all servers have exactly the same, or virtually the same, database and simply handle multiple requests or take actions for users in parallel, possibly checkpointing with each other as each action is taken, and
- Loosely-consistent architectures, in which the servers have mostly the same information or at least information in the same domain, but they do not try to enforce a particularly strong and consistent world view among themselves.

The choice of tight or loose consistency is generally a function of the operations being supported by the servers.

Advantages:

- These architectures are handy when it is relatively simple to maintain database consistency between servers (for example, if user requests or actions taken on their behalf do not side-effect the database, then its consistency is easier to maintain).
- Load-balancing is fairly simple, and extra hosts can be added incrementally to accommodate increases in load.
- The servers may be geographically distributed to improve either network load-balancing, time zone load-balancing, or fault-tolerance.

Disadvantages:

- If the algorithm requires tight consistency, the requisite interserver communications costs can eventually come to dominate the computation.
- Even loosely-consistent servers will probably still suffer from roughly quadratic growth in load with the number of users. This implies that, to keep up with even linear user growth, a quadratically-increasing number of servers must be put online; keeping up with typical exponential growth, of course, is much harder.

5.5.3 Multiple, non-mirrored servers

These types of agent architectures can fairly trivially be divided into:

- Those that know about each other, and

- Those that probably don't.

The Web itself is an example of this architecture; each server does not need to know about each other, and, in general, do not mirror each other. Few agent architectures seem to be designed in this way, however, except in the limit of the same agent simply being run in multiple instantiations in different information domains.

Advantages:

- Consistency is easy to achieve.
- Load sharing may be implemented as in the mirroring case above.

Disadvantages:

- Similar to mirrored servers, though the disadvantage of maintaining consistency is eliminated.
- Load growth may still be a problem.
- It may be difficult to find all servers if the algorithm demands it, since the lack of mirroring means servers may tend to fall out of touch with each other.

5.5.4 Totally distributed peers

As in the case above of multiple, non-mirrored servers, totally-distributed peers can be divided into:

- Those that know about each other
- Those that probably don't

This approach resembles an ALife system more than the approaches above, and deviates most radically from typical client/server or centralized-system approaches.

Advantages:

- Can probably be scaled up to accommodate loading easily, because servers are also peers and probably associate close to 1-to-1 with the user base.

- No central point of either failure or compromise.

Disadvantages:

- Coordination between the peers becomes much more difficult.
- Algorithms that require global consistency are probably impossible to achieve with acceptable performance.
- It may be difficult to keep all agents at similar software revision levels.

5.6 Applications

Many existing and potential industrial and commercial applications for DAI and multiagent systems are described in the literature [7]. Basically following examples of such applications are:

- Electronic commerce and electronic markets, where "buyer" and "seller" agents purchase and sell goods on behalf of their users.
- Real-time monitoring and management of telecommunication networks, where agents are responsible, e.g., for call forwarding and signal switching and transmission.
- Modelling and optimization of in-house, in-town, national- or worldwide transportation systems, where agents represent, e.g., the transportation vehicles or the goods or customers to be transported.
- Information handling in information environments like the Internat, where multiple agents are responsible, e.g., for information filtering and gathering.
- Improving the flow of urban or air traffic, where agents are responsible for appropriately interpreting data arising at different sensor stations.
- Automated meeting scheduling, where agents act on behalf of their users to fix meeting details like location, time and agenda.
- Optimization of industrial manufacturing and production processes like shopfloor scheduling or supply chain management, where agents represent, e.g., different workcells or whole enterprises.

- Analysis of business processes within or between enterprises, where agents represent the people or distinct departments involved in these processes in different stage and at different levels.

- Electronic entertainment and interactive, virtual reality-based computer games, where, e.g., animated agents equipped with different characters play against each other or against humans.

- Design and re-engineering of information- and control-flow patterns in large-scale natural, technical, and hybrid organizations, where agents represent the entities responsible for these patterns.

- Investigation of social aspects of intelligence and simulation of complex social phenomena such as the evolution of roles, norms, and organizational structures, where agents take on the role of the members of the natural societies under consideration.

5.7 Conclusion

Distributed intelligent agents have the potential to play a significant role in the future of software engineering. Further research is needed to develop the basis and techniques for societies of computational agents that execute in open environments for indefinite periods [8].

Distributed planning has a variety of reasonable well-studied tools and techniques in it repertoire. One of the important challenges to the field is in characterizing these tools and understanding where and when to apply each. Until many of the assumed context and semantics for the plans are unveiled, the goal of having heterogeneous plan generation and plan execution agents work together is likely to remain elusive. The field of distributed problem solving is even more wide open, because the characterization of a 'problem' is that much broader. Representations and general-purpose strategies for distributed problem solving are thus even more elusive. Distributed problem solving and planning strategy has still more 'art' to it than we like to see in an engineering discipline [9].

Although real-time search provides an attractive framework for resource-bounded problem solving, the behavior of the problem solver is not rational enough for autonomous agents: the problem solver tends to perform superfluous actions before attaining the goal; the problem solver cannot utilize and improve previous experiments; the problem solver cannot adapt to the dy-

namically changing goals; and the problem solver cannot cooperatively solve problems with other problem solvers [10].

In the future, systems will increasingly be designed, built, and operated in a distributed manner. A large number of systems will be used by multiple real-world parties. The problem of coordinating these parties and avoiding manipulation cannot be tackled by technological or economic methods alone. Instead, the successful solutions are likely to emerge from a deep understanding and careful hybridization of both [11].

Centralized systems are failing for two simple reasons: They cannot scale, and they don't reflect the real world of people [12].

Decentralized approaches often seem impractical, but they work in practice. The Internet itself is a prime example–it works because the content, the domain name system and the routers are radically distributed. But it's the human element that is really driving the pressure for decentralized solutions. This shouldn't be too surprising. Biological phenomena like the human body and the global biosphere have had billions of years to evolve, and they are the most complex decentralized systems we encounter. Decentralization is neither automatic nor absolute. The most decentralized system doesn't always win. The challenge is to find the equilibrium points–the optimum group sizes, the viable models and the appropriate social compromises.

Bibliography

1. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 1.

2. http://webopedia.internet.com/Term/a/artificialintelligence.html

3. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 2–5.

4. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 8–9.

5. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 79–81.

6. http://foner.www.media.mit.edu/people/foner/Essays/Agent-Coordination/degree-of-decentralization.html

7. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 6–7.

8. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 28–73.

9. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 121–158.

10. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 165–196.

11. Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* The MIT Press, 1999, pp. 201–251.

12. http://192.246.69.113/archives/000010.html