

# Multi Agent System for Negotiation and Decision Support

Zoltán BALOGH, Michal LACLAVÍK, Ladislav HLUCHÝ  
Institute of Informatics, Slovak Academy of Sciences, Bratislava  
balogh.ui@savba.sk, laclavik.ui@savba.sk, hluchy.ui@savba.sk  
<http://ups.savba.sk/>

## Abstract

In this paper we analyse multi-agent systems. Recommendation of techniques for negotiation and decision support of mobile agents are presented. We support our theory by providing a simple example of an intelligent multi-agent negotiation and decision making system. Example is developed by Java IBM Aglet multi-agent system.

## 1 Introduction

Agent-based systems technology has become a new paradigm for conceptualizing, designing, and implementing software systems. Agents are sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems. Increasingly, however, applications require multiple agents that can work together. A Multi-Agent System (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver [11]. Mobile agents are a beneficial technology for the creation of distributed systems. The reasons supporting this statement, are as follows [1]:

- *Agents reduce the network load.* Mobile agents allow to package a conversation and dispatch it to a destination host, where the interaction can take place locally.
- *Agents overcome network latency.* Mobile agents offer a solution to problem of unacceptable latencies in critical real-time systems, because they can be dispatched from a central controller to act locally and directly execute the controller's directions.

- *Agents encapsulate protocols.* Communication protocols often become a legacy problem. Mobile agents can move to remote hosts to establish channels based on proprietary protocols.
- *Agents execute asynchronously and autonomously.* Mobile devices must often rely on expensive or fragile network connections. Embedding tasks into mobile agents, which can then be dispatched into the network, solves this problem.
- *Agents adopt dynamically,* by the ability to sense their environment and react autonomously to changes.
- *Agents are naturally heterogenous.* Because mobile agents are generally dependent only on their execution environment, they provide optimal conditions for system integration.
- *Agents are robust and fault-tolerant,* because of their ability to react dynamically to unfavorable situations and events.

## 2 Research Infrastructure for the Model

There are many infrastructures for developing more or less complex distributed multiagent systems. The Aglets Software Development Kit (SDK) is an environment

for programming mobile autonomous Internet agents in Java. An Aglet (Light Weight Agent) is a Java object that can move from one host on the network to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built-in security mechanism makes it safe for a computer to host untrusted aglets [14].

For implementing the model proposed in this article we chose IBM's Aglets SDK. The ASDK uses Java to implement agents. The following facts suspected Java as an ideal programming language suitable for our purposes:

- Java supports secure migration of classes.
- Interfaces to systems "speaking" KQML are available for Java.
- Java is platform independent.
- SQL interfaces to databases are implemented.
- There are development environments available for Java.

### **3 Negotiation and Decision Support**

#### **3.1 Present Stage of Negotiation**

Negotiation theory was first used in game theory. Negotiation in game theory is different from real life. The game software does not have to learn to make good decisions. The game has already all needed information for decision making. Still in game theory, we can find two good ideas:

- In game theory, agents have knowledge of other agent negotiation and decision making algorithm.
- Game agents have all needed information for negotiating on a certain place and they can use common knowledge instead of its own.

Auctioning is most popular and mostly used negotiation strategy. Usually agents are auctioning on goods or services. User creates agent with certain conditions, values, and bidding strategies and agent is bidding in auction instead of the user.

In this paper, we present Cut cake algorithm that can be also used as Auctioning algorithm with some modifications. Nomad Auctioning System or eBay.com site can be quoted as good examples of these systems.

Negotiation Agents can be also used to improve regular person-to-person negotiation. InterNeg project is good example. [16]. These way agents can be used to search, sort, choose, or recommend some information, solution, but final decision is left to the users. Negotiation does not have to end-up with rejecting or accepting. The negotiation result can be data for next negotiation. In this case, agents work as a good secretary.

Communication is always negotiating. All of multi-agent systems have to use kind of communication-negotiation protocol. Negotiation needs some communication protocol for agents to understand each other. There are many communications languages and protocols. Many agents systems are building its own ones. KQML standard is promising technology in this area.

Agents are promising technology for Market Places. The most important role is role of negotiation, which is taking place of people negotiation when buying and selling goods. Several projects of Market Places have been developed. One of them KABASH, [15] is a multi-agent electronic marketplace that proposes to transform the online shopping experience by leveraging the unique features of the Internet and software agent technologies. Aglets Market

Place is newer market place[14]. It is a based on Aglets mobile agent technology.

*Possible negotiation problems and solutions:*

- Decision Making
- Learning
- Coordination – cooperation, moving from place to place
- Negotiation – cooperation, competition
- Similarity to human negotiation
- Fair and envy-free

Agent Negotiation is always based on conditions (price > offer, time = time of session etc.). Real life negotiation is not always like this. For negotiating strategies non-condition decision making is required. We could possibly use neural networks or some genetic based algorithms. Here we can see the problem of storing and sorting the information. It is very hard for an agent to learn as neural networks if it has to carry the information with itself. It is better to create some negotiation-decision centers. The agent then passes only its conditions and states and the negotiation center will provide the negotiation.

Fair negotiation is also a problem. It can be solved by knowledge negotiating algorithm of other agent. In the negotiating center, this can be solved by increasing knowledge of center by each negotiating. Envy-free and also fair is the Cut Cake algorithm, which is also described below.

Cooperative negotiation – cooperation on projects, meetings, etc. Goals: - we know here that we can give our (e.g. Timetable data) data to other agent and he will give ours. Agents will agree on some time if possible. If we have some profit function, agents can put profit function together – the best solution is found.

Non-cooperative negotiation: Problems with same conditions of profit for all negotiators.

### 3.2 Decision Making and Rationality

According to Doyle [5] judgements of human rationality commonly involve several different conceptions of rationality, including a *logical* conception used to judge thoughts, and an *economic* one used to judge actions or choices. Intelligence involves both perception and action. In most cases, actions are not determined by the agent's situation, but instead involve choices to do one thing rather than another. Thus, both thinking and choice are central operations in thinking. The fundamental issue in the theory of economic rationality is choice among alternatives. Economic rationality means making „good“ choices, where goodness is determined by how well choices accord with the agent's preferences among alternatives.

#### *Preferences*

Preference is the fundamental concept of economic rationality. We write  $A \succ B$  to mean that agent prefers  $B$  to  $A$ , and  $A \sim B$  to mean that the agent is indifferent between the two alternatives. We also write  $A \not\succeq B$  to mean that  $A \succ B$  does not hold and  $A \not\sim B$  that either  $A \sim B$  or  $A \succ B$ . The collection of all these comparisons constitutes the agent's set of preferences. These preferences may change over time. Rational agents choose maximally preferred alternatives. If  $\alpha = \{A_1, A_2, \dots, A_n\}$  is the set of alternatives, then  $A_i$  is a rational choice from among these alternatives just in case  $A_i \succ A_j$  for every  $A_j \in \alpha$ . There may be several rational choices, or none at all if the set of preferences is inconsistent or if the set of alternatives is empty or infinite.

The theory does not require that a rational agent explicitly calculates or computes the maximality of its choices, only that the agent chooses alternatives that are in fact maximal according to its preferences. The theory requires, as a minimum basis for rationality, that strict preference is a strict partial order, indifference is an equivalence

relation (transitive and asymmetric), and any two alternatives are either indifferent or one is preferred to the other, but not both at the same time. These separate requirements on preference and indifference may be put formally, for all alternatives  $A, B$  and  $C$ :

1. Either  $A \succsim B$  or  $B \succsim A$ ,  
(completeness)
2. If  $A \succsim B$ , then  $A \not\succeq B$ ,  
(consistency)
3. If  $A \succsim B$  and  $B \succsim C$ , then  $A \succsim C$   
(transitivity).

The rationality constraints imply that we may represent the set of preferences by means of a numerical utility function  $u$ , which ranks the alternatives according to degrees of desirability, so that  $u(A) < u(B)$  whenever  $A \prec B$  and  $u(A) = u(B)$  whenever  $A \sim B$ . By working with utility functions instead of sets of preferences, we may speak of rational choice as choosing so as to maximize utility. The same set of preferences may be presented by many utility functions, as any strictly increasing transformation of a utility function will provide the same choices under maximalization. The distinction between the cost or values of something and its utility or disutility is one of the great strengths of the theory of economic rationality. The cost of some alternative says nothing about the value or benefits received from it, and neither cost nor benefit need be identical with the utility of the alternative if the risk posed by the action diminishes (or increases) its attractiveness to the agent. The utility of an action is usually some function of the cost, benefit, risk and other properties of the action.

### Decision Theory

Compared with the basic theory, decision theory adds probability measures  $p_A$  which indicate the likelihood of each possible outcome for each alternative  $A$ . Decision theory supposes that the agent does not know the actual situation, but does have

beliefs or expectations about the consequences of choice in different states. Decision theory also strengthens the notion of utility from an *ordinal* utility function  $u$  to a *cardinal* utility function  $U$ , which inputs different values to each possible outcome. Ordinal utility functions use numeric values simply as ways of ranking the alternatives in a linear order. Amounts of cardinal utility can be added and subtracted to produce other amounts of utility. This makes it possible to combine the utilities foreseen in different possible outcomes of  $A$  into the expected utility  $\hat{U}(A)$ , defined to be the utility of all possible outcomes weighted by their probability of occurrence, or formally,

$$\hat{U}(A) \stackrel{\text{def}}{=} \sum_S p_A(S)U(S),$$

where the sum ranges over all possible situations or states of nature under discussion. The decision-theoretic definition of preference is

$$A \succsim B \text{ if and only if } \hat{U}(A) \leq \hat{U}(B).$$

Like the theory of preference, the assumptions of decision theory can be also formulated qualitatively [5].

### 3.3 Negotiation Model of Goods and Services

This model is fair and envy-free. In this model, we can see how real life solution would be useful in agent negotiation. One person divides the cake in two, and the other chooses the first piece. This is fair in that each believes he or she received at least a half, and envy-free in that neither would wish to trade. If a third person will come, each of those two will cut his/her portion into three parts and the third person will take one piece from each one. This protocol succeeds even when participants disagree about the portion's value [7].

Let's bring this protocol into Agents world. First Agent can use the cake-cutting algorithm when negotiating for resources.

For example, two agents sharing CPU time might also wish to complete their computation as soon as possible. One agent might select the sizes of the processor's time slices, while the other would have first choice of the resulting slices. Similar negotiation could occur over bandwidth, relative cache size, locks on databases, storage space in a file system, or task decomposition and distribution. This algorithm can be applied not only on two agents but on N agents as well. When n+1 agents come to negotiate, each of n agents will divide its sources into n+1 parts. Agent n+1 will choose one slice from each agent.

*Formal Description of Cut Cake Algorithm*

$\langle A, S, F, D \rangle$

$A$  – A set of Agents for negotiation.

$S$  – A set of sources for negotiation.

At the beginning, one agent holds this set.  $S_i$  is the source held by  $A_i$  agent.

$F$  – A set of  $f_i$  functions  $f_i \in F$ , and  $f_i$  is cutting function or algorithm of agent  $A_i$ .

$D$  - A set of  $d_i$  functions  $d_i \in D$ , and  $d_i$  is decision function or algorithm of agent  $A_i$ . This function is used for decision making.

#### 4 Implementation of Goods and Services Algorithm

In a classical multi-agent system, each autonomous agent must be able to decide how to behave in various situations. Coordination, negotiation, decision-making, and learning, like other agent's activities, create a huge demand on agent's problem solving abilities. Creating greater demand means making agents more complicated. In such situations, mobile agents must carry bigger code segments, what is reflected in heavier network load and consequently inefficient resource exploitation. Therefore, we think that creating specialized agents is a good solution to lighten the demand on

individual agents. Reasonable specialization would make it possible for an agent to facilitate services of other agents and therefore be more concerned with tasks delegated to him.

Class	Description
NegCenter	Represents the negotiation center. It is a subclass of the main Aglet class. Agents can access it with KQML requests for negotiating. Class stores information about services into MySQL database via a JDBC interface. The following information is stored: name, category, description, minimum quantity, maximum quantity, value (price). This information is stored when agent sends a bid to negotiation center. It is used when agent sends offer to center. It is also sorted by prices and possibilities and passed to agent.
AgentNeg	Subclass of Aglets and JKQML.Supports KQML requests and NegCenter specific communication functions.
AgentBidder	Subclass of AgentNeg class. Can be used as agent which is bidding some services (seller).
AgentOffer	Subclass of AgentNeg class. Can be used as agent which is offering some services (buyer).

*Tab. 1. Implemented classes*

Our main goal is to create an agent, nested into a multi-agent system, which would handle negotiation and support decision making of an agent domain. The main entity in our model, called negotiation center, collects all the information needed for effective decision making (Fig. 1). Exchange of detailed specifications and information is done by means of negotiation. Concentration of all information about offered and demanded goods and services creates a framework for effective functioning of agent domains. In addition,

benefits of adding new features and abilities to the negotiation center are shared by every member in a domain. Detailed description of each class of our model is given in Tab. 1.

Negotiation center uses decision algorithms to rationally apportion goods and services into parts with equal utilities. In order to enable the system to compute utilities, negotiation center needs to know the utility functions of particular goods and services. Individual utilities are functions of elements such as price, amount, time, etc. On the other hand, client agents need to make the best choice among alternatives offered by the negotiation center. We suppose that each client agent aims at maximizing the total agent's utility while choices must accord with the agent's preferences.

other multi-agent platforms. Using standard protocols makes the model suitable for verification in real commercial systems.

## 5 Conclusion and Future Work

In the article, we provide formal description of algorithms for negotiation and decision support. We also propose a reference model of a multi-agent system. The main entity in our model is represented by a negotiation center, which assists other agents with negotiation and decision making inside an agent domain.

By implementing our model we want to prove legitimation of specialized agents, as well as manifest the benefits the rational specialization of agents might bring. Our future work will be primarily concerned with improving the architecture of our model, as

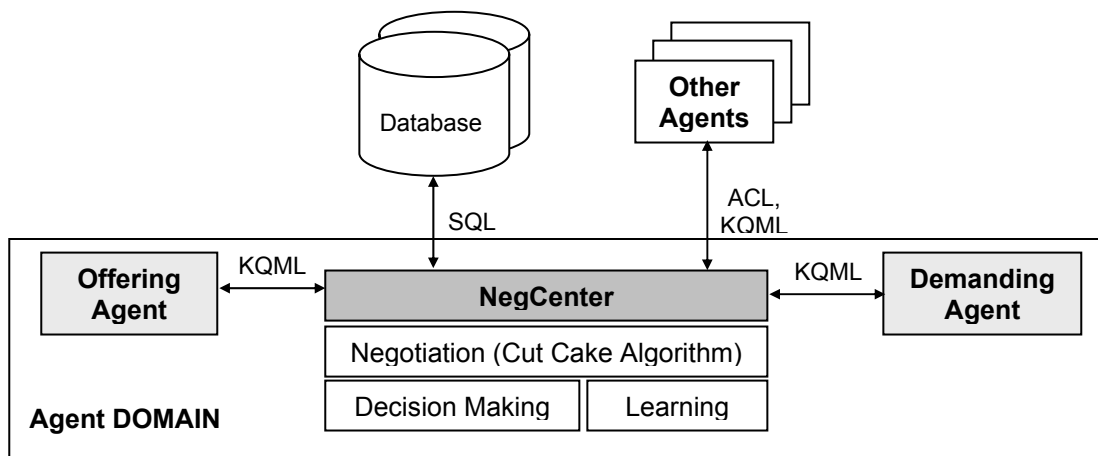


Fig. 1. Model

The relative disadvantage of our model is that each agent in the agent domain must truly believe that the specialized agent which facilitates negotiation and decision support for him is fair. To ensure fair behavior of our negotiation center, we utilize the Cut Cake algorithm described above. Negotiation and resource distribution by this algorithm is fast, fair and efficient.

We base the inter-agent communication on KQML, which also ensures standard message interchange with

well as accommodating algorithms for multi-agent environments.

## References

- [1] Bigus, J.: Constructing Intelligent Agents with Java. Willey Computer Publishing, Canada, 1997.
- [2] Chuang, T., Yadav, S., B.: An Agent-Based Architecture Of An Adaptive Decision Support System, 1997.

- [3] Cockayne, W., R., Zyda, M.: Mobile Agents, Manning Publication co., USA, 1997.
- [4] Cremonini, M., Omicini A., Zamboneli, F.: Ruling Agent Motion in Structured Enviroments, HPCN Europe, 2000.
- [5] DOYLE, J.: Rationality and its Roles in Reasoning. In Computational Intelligence, Vol. 8, No. 2 (May 1992), pp. 376-409.
- [6] Genesereth, M., R., Ketchpel, S. P.: Software Agents.
- [7] Huhns, M., N., Malhotra, A., K.: Negotiating for Goods and Services, IEEE Internet Computing, July-August 1999.
- [8] Labrou, Y., Finin, T.: A Proposal for a new KQML Specification, TR CS-97-03, available through the Stanford University Computer Science Department, 1997.
- [9] Lange, D., B., Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, Canada, 1998.
- [10] Zeng, D., Sycara, K.: How Can Agent Learn to Negotiate? In: Intelligent agents III, ECAI '96 Workshop, pp.233-244, 1996.
- [11] Intelligent Software Agents.  
[<http://www.cs.cmu.edu/~softagents/>]
- [12] Knowbot System Software  
[<http://www.cnri.reston.va.us/home/koe/>]
- [13] The Open Agent Architecture  
[<http://www.ai.sri.com/~oaa/>]
- [14] Agents Software Development Kit Home [<http://www.trl.ibm.co.jp/aglets/>]
- [15] KABASH  
[<http://ecommerce.media.mit.edu/kasbah/>]
- [16] FOR AND ABOUT NEGOTIATIONS  
[<http://www.InterNeg.org/>]