

## Software Agents: Quality, Complexity and Uncertainty Issues

Behrouz Homayoun Far  
Faculty of Engineering, University of Calgary  
far@enel.ucalgary.ca

### Abstract

*In software engineering community there is an increasing effort of design and development of multi-agent systems (MAS). Among several issues emerging from this initiative, complexity, quality and uncertainty issues have not yet received much attention. The main factors affecting quality of MAS are complexity and knowledgeability. Complexity of MAS can be defined in terms of structural and algorithmic complexity in either objective or subjective way. Knowledgeability of MAS can be defined in terms of problem solving and cognitive capabilities and the ability to cope with interactions, such as cooperation, coordination and competition. In this paper we define and address the quality and complexity issues of MAS and define metrics to measure the quality and complexity. The metrics are used for devising a candidate set of agents for MAS design. Furthermore, we address the knowledgeability of MAS and devise models and techniques to cope with uncertainty in competitive situations.*

### 1. Introduction

Nowadays, an increasing number of software projects are being revised and restructured in terms of multi-agent systems (MAS). Software agents are considered as a new experimental embodiment of computer programs and are being advocated as a next generation model for engineering complex, heterogeneous, scalable, open, distributed and networked systems. However, agent system development is currently dominated by informal guidelines, heuristics and inspirations rather than formal principles and well-defined engineering techniques. There are some ongoing initiatives by The Foundation for Intelligent Physical Agents (FIPA) (<http://www.fipa.org>) and some other institutions to produce software guidelines and standards for heterogeneous, interacting agents and agent-based

systems. However, such initiatives fall short to address the quality and complexity issues explicitly.

### 2. MAS quality

Quality for software systems can be defined in terms of conformance to requirement [3] or fitness for use [8]. In the former, the requirements should be clearly stated and the product must conform to it. Any deviation from the requirements is regarded as a defect. Therefore, a good quality product may contain fewer bugs. The latter, puts emphasis on meeting user's needs. Therefore, a good quality product provides better user satisfaction.

Conventional software quality models, such as CUPRIMDA (based on 8 fitness parameters: capability, usability, performance, reliability, installability, maintainability, documentation and availability) [9], Boehm's [2], McCall's [12], and ISO 9126 (based on 6 characteristic factors: functionality, reliability, usability, efficiency, maintainability and portability) address quality in terms of a few quality attributes (or factors) and criteria (or intermediate and primitive constructs) to represent the attributes. The criteria (or primitive constructs) are later mapped to actual metrics.

Based on the above discussion, quality in MAS can be examined from various viewpoints such as:

- Conformance: conformance to customers' requirements; conformance to standards;
- Development process quality: requirement, design, implementation, test and maintenance quality;
- End-product quality: reliability, usability and availability;
- Relativity: advantage over similar products;

The two main factors affecting quality of MAS from both the customer and developers' point of view, are *complexity* and *knowledgeability*. Complexity of MAS can be defined in terms of structural and algorithmic complexity in either objective or subjective way. Knowledgeability of MAS can be defined in terms

of problem solving and cognitive capabilities and the ability to cope with interactive scenarios, such as cooperation, coordination and competition. Complexity and knowledgeability issues are discussed in the next two sections.

### 3. Complexity in MAS

In conventional software systems complexity is structural in nature. As the system evolves new components or functions may be added to the system. By doing so, the structure of the software may deteriorate to the extent that major effort is needed to maintain its consistency and conformity with the requirements.

On the other hand, complexity of MAS is both *structural* and *algorithmic*. They both can be defined in either objective or subjective way. We elaborate on this in the following subsections.

#### 3.1. Structural complexity

A main complexity component in MAS is structural because new agents may be added to the system or new functions, program modules or packages may be added to the existing agents. The MAS architecture is the primary artifact for conceptualizing, constructing, managing, and evolving the system under development.

It is difficult to draw a sharp line between software design and its architecture. Software architecture is a level of design concerned with issues beyond the computation. Architectural issues include strategic decisions upon:

- Structural issues including gross organization and global control structure.
- Selection among design alternatives.
- Assignment of functionality to constituent agents.
- Composition of constituent agents.
- Protocols for communication, synchronization, etc.
- Physical distribution.
- Scaling and performance.

Hierarchical decomposition is a major method of handling complexity in conventional software analysis and design, assuming that the final product shall have the hierarchical architecture. Unfortunately, hierarchical decomposition cannot be used directly in MAS system development due to the facts that the MAS architecture may not necessarily be hierarchical and MAS analysis and design is not essentially top-down or bottom-up. That is, the participating agents of the MAS cannot be defined at the outset in a hierarchical way. The interactions of the MAS system with the outside world, i.e., use case models, usually come first and then the architectural pattern and participating agents may be

decided upon. This is equivalent to moving up the hierarchy. Defining detailed design for each agent is equivalent to moving down the hierarchy.

An architectural pattern expresses a fundamental structural organization schema for the MAS systems. It provides a set of predefined agents, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them. The most popular architectural patterns for MAS are:

- **Layers:** application is decomposed into different levels of abstraction, such as application layer, business specific layer, middleware layer, etc., and each constituent agent may belong to one layer only.
- **Blackboard:** independent specialized agents collaborate to derive a solution, working on a common data structure called blackboard.

The architectural pattern may be extended to devise the internal architecture of each constituent agent. Common internal architectural patterns are:

- **Model-view-controller (M-V-C):** application is divided into three partitions. The *model*, which is the business rules and underlying data; the *view*, which is how information is displayed to the user; and the *controllers*, which process the user input.
- **Reasoning-communication-documentation engine (R-C-D):** the application is composed of three processing engines. The *reasoning engine* to process the basic business logic; the *communication engine* to manage messages to and from the outside world; and the *documentation engine* to manage data internally [4].

In MAS the relationships among the agents are dynamic. (If not, the system can be developed in a much easier way of using *pipes and filters* architecture, in which the data is processed in streams that flow through pipes from filter to filter.) Two kinds of dynamic relationships can be devised: *interactions* among subsystems and *intra-actions* within subsystems. Interactions are between an agent and its outer environment and manifested by the messages sent, received (in case of cooperation and coordination) and perceived (in case of competition). Intra-actions are the characteristics of the agent's inner environment. Contemporary software engineering techniques can manage the intra-actions using decomposition and abstraction techniques and interactions using RPC, RMI, etc.

#### 3.2. Algorithmic complexity

Algorithmic complexity stands for the mechanisms for knowledge processing and knowledge sharing as well as the ability to engage with the other agents in cooperative, coordinative and competitive tasks.

### 3.3. MAS complexity metrics

**3.3.1. Subjective metrics.** Subjective complexity accounts for the way that a human user evaluates the complexity of the agent system. A modified version of *Function Point (FP)* [1], that accounts for algorithmic complexity can be used. For each participant agent, the parameters involved in the model are: External inputs ( $N_i$ ) and external outputs ( $N_o$ ), external inquiries ( $N_q$ ), external interfaces ( $N_{ef}$ ), internal data structures ( $N_{if}$ ), algorithmic complexity ( $N_m$ ) and knowledge complexity factor ( $N_k$ ). The algorithmic complexity ( $N_m$ ) factor is the sum of three Boolean variables stating whether cooperative, coordinative and competitive mechanisms are implemented or not ( $0 \leq N_m \leq 3$ ). The knowledge complexity factor ( $N_k$ ) has a value between 0 and 5 depending whether the agent has a knowledge-base and whether the knowledge-base is sharable or is based on a shared ontology.

$$UF_eC = 4N_i + 5N_o + 4N_q + 7N_{ef} + 7N_{if} + 10N_m + 6N_k$$

The adjusted MAS function point (MAS-FP) is derived by multiplying  $UF_eC$  with the subjective assessment of technical complexity, the  $TCF$  factor [1]. The overall complexity of the MAS will be the mean of the adjusted feature points of its constituent agents.

**3.3.2. Objective metrics.** Objective complexity accounts for complexity as an internal property of the agent-based system. If the MAS system is nearly-decomposable, the cyclomatic complexity [11] metrics can be used. Complexity of the MAS is the sum of cyclomatic complexities of its constituent agents. As a measure for nearly-decomposability, the communicative cohesion metrics can be examined. The communicative cohesion metrics ( $CCM$ ) for an agent  $g_i$  is defined in terms of the ratio of internal relationships (interactions) to the total number of relationships (i.e., sum of interactions and intra-actions).

$$CCM(g_i) = \frac{R_{\text{internal}}}{R_{\text{internal}} + R_{\text{external}}}$$

The  $CCM$  for the MAS is the statistical mean of  $CCM$  of its constituent agents. Systems with  $CCM \geq 0.91$  are usually considered to be nearly-decomposable.

In this research, we identify two types of organizational relationships: *signal level* and *symbol level* relationships. *Signal level* accounts for dynamic message passing. At this level, messages between two communicating agents are interpreted via ascribing the same meaning to the constants used in the messages. In this way, mutual understanding of the domain constants

before further message passing is guaranteed. *Symbol level* relationships, on the other hand, account for dynamic knowledge sharing. The internal and external relationships in  $CCM$  account for signal level relations only.

### 3.4. Application

The first step in design of a MAS system is to sketch the use cases and then identify constituent agents using the use cases and architectural patterns. The problem is how to devise the constituent agents. The MAS complexity metrics can be used for decomposing the problem based on function/ input/ output into an organization of agents and refining this list. First, the target  $CCM$  and  $UF_eC$  is set and decomposition is performed to devise a tentative set of agents with  $CCM$  greater than the target value. Then the  $UF_eC$  is measured for each agent and those with higher  $UF_eC$  value will be the target for further decomposition. These steps are repeated until all the agents have satisfactory  $CCM$  and  $UF_eC$ .

## 4. Knowledgeability in MAS

Traditional software engineering can handle *data* and *information*. *Data* is defined as a sequence of quantified or quantifiable symbols. *Information* is about taking data and putting it into a meaningful pattern. *Knowledge* is the ability to use that information. Since knowledge is prerequisite to wisdom, one always wants more data and information. Knowledgeability is an integral part of MAS paradigm and agents in order to interact and work proactively must be knowledgeable in their area of expertise. Knowledgeability can be defined in terms of *cognitive capabilities* and *interactions*. The following subsections elaborate this.

### 4.1. Cognitive capabilities

Three main capabilities of agents in MAS are *representing*, *using* and *sharing* the knowledge.

Symbol structure (SS) is used to model individual agent's knowledge structure. SS is a finite connected multi-layer bipartite graph. There are two kinds of nodes in each layer of SS: concepts (c) and relations (r). One source of difficulty when processing concepts, is distinguishing a concept at various levels of abstraction. Function type is defined to ease such differentiation. The function type maps concepts and relations onto a set T. The elements of T are called type labels. In this way, moving from any level of abstraction to another is supported [5].

The type hierarchy is a partial ordering defined over the set of type labels.

We can mention that SS is semantically richer than semantic networks because both the concepts and relations are augmented with types. Although there are other methods to represent the private knowledge and domain ontology such as *conceptual graphs* [13], they do not explicitly define mechanisms for combining and using the private knowledge together with the agents' profile of actions in the MAS environment.

Ability to use the knowledge can be realized by having a knowledge-base in the form of SS and mechanisms for problem solving using the knowledge base.

Finally, ability to share the knowledge depends on ontologies for the domain and task. Mechanisms for using and sharing are presented in [6].

## 4.2. Interaction in MAS

Basic agents' interactions are cooperation, coordination and competition.

- Cooperation: Cooperation is revealing an agent's goal and the knowledge behind it to the other party. In cooperation both agents have a common goals.
- Coordination: Coordination is revealing an agent's goals and the knowledge behind it to the other party. In coordination, agents have separate goals.
- Loose Competition: Loose competition is revealing only an agent's goals but masking the knowledge behind it to the other party.
- Strict Competition: Strict competition is neither revealing an agent's goals nor the knowledge behind it to the other party.

Almost all of the proposed MAS frameworks and guideline assume that the agents are engaged in cooperative and coordinative tasks only. Figure 1 shows decision making mechanism based on agents' interaction.

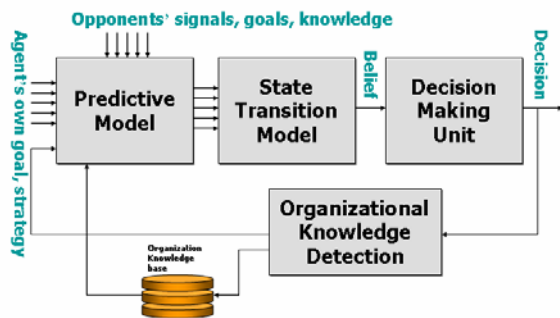


Figure 1. Decision making mechanism

## 5. Uncertainty in MAS

In this section, we consider multi-agent interaction under competitive and uncertain environments.

Information gained through agents' behaviour, i.e., *signals* might be incomplete in competitive environments since agents may try to hide their strategies. Agents in competitive environments, must make decisions under uncertainty, predict environment's parameters, predict other agents' future moves, and successfully explain self and the other agents' actions. The crucial factors are both the amount and specification of the information. Any lack of information and/or noise affects the quality of decisions, the moves to be performed and conclusion to be devised. In this section, we propose a game theoretic approach to handle this case.

### 5.1. Overview of multi-agent competitive environment

Fig. 2 shows the outline of agent competition. The process for deciding competitive strategy includes following steps. First, each agent tries to predict opponents' strategy. Second, the agent chooses the best response strategy based on previous predictions. And finally, each agent will get a payoff by using a utility function.



Figure 2. Overview of agents' competition

From the decision making viewpoint, since the amount of payoff in extended games is influenced by the opponent's moves, exact predictions of the other agent's strategies is crucial to guarantee a stable payoff. Information about opponents' movements is *uncertain* because they may try to hide their strategies and the knowledge behind it as much as possible. Therefore, in order to cope with multi-agent competitive environments, modeling and formalization of strategic decision making and uncertainty management method is required. To meet this requirement, the work presented here suggests modeling an incomplete game theoretical based decision making method for competitive agents.

### 5.2. Modeling competitive environment

One cannot exactly predict some agent's strategy because of lack of knowledge on opponent's *preference* since strategies are selected based upon each agent's

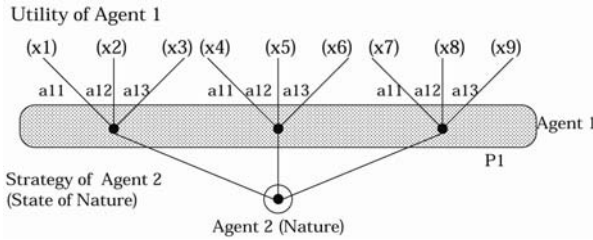
preference relations. In order to model such situation, we divide (opponent) agents into following two kinds:

- Agents whose preference relations are exactly known.
- Agents whose preference relations are not known.

As for first, we treat them based on normal game theoretic approach. As for second, we regard these agents as *natural* objects and their strategies as *state of nature* and treat them by lumping up all of the natural objects as uncertainty. Following this principle, we define a multi-agent competitive environment as:

$$\Gamma = \{A, N, X, \{S_i, \succ_i, P_i\}_{i \in A}\}$$

Where  $A$  is the set of agents,  $N$  is the set of states of nature;  $X$  is the outcome of competition and it is defines as:  $S_1 \times \dots \times S_n \times N \rightarrow X$ ; where,  $S_i$  is the set of strategies of agent  $i$ ;  $\succ_i$  is the preference relation of agent  $i$ ; and  $P_i$  is the information partition over the state of nature of agent  $i$ . It is represented by extensive form of a game as shown in Fig. 3. This is a simple but illustrative example of agents' competition model (*agent 1* versus *agent 2*).



**Figure 3. Example of competition model**

In this example, we consider that *agent 1* doesn't know the preference relation of *agent 2* and thus, *agent 1* is uncertain about which strategy *agent 2* might adopt. Here,  $P_1$  is an information partition of *agent 1* and it is not sure which nodes he stays in (left, right or center) within  $P_1$ . Under this uncertain environment, *agent 1* must decide which strategy to adopt so as to optimize its utility. In this case, *agent 1* assigns its *belief* (which never violates the probabilistic principle) over the state of nature and adopts the strategy which maximizes the expected utility. If all the agents decide upon strategy in this way, there is a possibility that it leads to social equilibria (Bayesian Perfect Nash Equilibria) [10]. A question which naturally arises here is how each agent assigns his belief autonomously. Below, we devise such decision making method by dividing uncertain environment into three levels.

### 5.3. Decision making in competitive environment

In a competitive environment, each agent encounters uncertainty generated by the lack of knowledge when devising optimal explanations to observable actions of other agents. Generally, certainty is divided into three levels according to the amount of information about the state of nature or given signal observed before choosing among several strategies [7].

- **Level-1: Decision Making under Certainty**  
The agent knows exactly what the state of nature will be. In this case, strategy is straightforward, it selects the strategy which maximizes its utility using traditional game theoretic approach.
- **Level-2: Decision Making under Risk**  
It is assumed that the agent is not sure what state of nature will be, but it has a probability distribution over the state of nature. In this case, the agent assigns known probability distribution as its belief and selects the strategy which maximizes expected utility. Below we propose a risk management method in order to reflect each agent's *attitude toward risk*.
- **Level-3: Decision Making under Uncertainty**  
In this level, we assume that the agent doesn't know anything about the state of nature except for that it is in some set,  $N = \{\omega_1, \omega_2, \dots, \omega_n\}$ . In this case, agent has to assign his belief without using probability distribution. Below we propose a new decision making and belief assignment which reflects agent's *degree of optimism*.

It should be noted that decision making under certainty (*Level-1*) is a special case of decision making under risk (*Level-2*) where probability distribution over states of nature is 1.0 (and variance is 0).

**5.3.1. Decision making under risk.** In case of decision making under risk, the agent naturally selects a strategy which maximizes its expected utility. Generally, utility function is decided by calculating expected value of cost and/or benefit. If expected values of two strategies are the same, these two strategies always become non-discriminateable. However, one cannot say that this decision rule is rational. This is because, even if the expected values are the same, when variance is large, risk of failure increases. Therefore, it is natural to consider that risk of failure influences decision making. Therefore in order to make decisions under the risk, we must reflect each agent's *attitude towards risk*. Generally, attitude towards risk is categorized into the following three types [7].

- *Risk prone*: In this case, agents prefer high risk-high return strategy rather than low risk-low return strategy.
- *Risk aversion*: In this case, agents prefer low risk low return strategy rather than high risk-high return strategy.
- *Risk neutral*: If expected value is the same, these strategies always become non-discriminateable.

In the field of economics, attitude toward risk is reflected by defining subjective probability strictly. But this is too complicated and computationally expensive to be implemented on artificial agents. Therefore, we use heuristic function that reflects the agent's attitude towards risk. The utility function is defined by:

$$u(x) = E(x) - \eta V(x)$$

Where  $x$  is a pure benefit when agent adopts some strategy,  $E(x)$  is a expected value when agent adopts some strategy,  $V(x)$  is a variance and  $\eta$  is a coefficient of degree of risk aversion taking values between  $-1$  and  $+1$ . If  $\eta$  is plus, the function  $u(x)$  becomes risk aversion because, the larger variance is (means the larger risk of failure is), the smaller utility becomes. Conversely, if  $\eta$  is minus, function  $u(x)$  represents risk prone because, the larger variance is, the larger the utility becomes. And if  $\eta$  is zero,  $u(x)$  represents risk neutral because,  $u(x)$  is equal to the expected value when the agents adopt some strategy.

Using this method, agents are allowed to select a strategy reflecting attitude toward risk and this simple representation can be easily implemented.

**5.3.2. Decision making under uncertainty.** In case that the agent has to decide upon the strategy under uncertainty, it has to assign its belief without using a probability distribution. According to psychologists, when human doesn't know probability distribution over uncertainty (such as state of nature), he/she decides upon action or strategy based on *degree of optimism*. Here we quantify each agent's *degree of optimism*.

In order to quantify degree of optimism, we use Ordered Weighted Averaging (OWA) operator [14]. OWA operator of dimension  $n$  is defined as the function  $F$  that has associated with a weighting vector  $\mathbf{W}$ ,

$$\mathbf{W} = [w_1, w_2, \dots, w_n]$$

such that,  $0 \leq w_j \leq 1$  and  $\sum_j w_j = 1$

and for any set of some value  $a_1, \dots, a_n$

$$F(a_1, a_2, \dots, a_n) = \sum_j w_j b_j$$

where  $b_j$  is the  $j^{th}$  largest element in the collection  $a_1, \dots, a_n$ .

Various semantics can be associated with the OWA aggregation procedure in this framework of decision making under uncertainty, such as viewing the OWA weights as a pseudo-probability distribution [15]. In particular we can view  $w_j$  as a kind of probability that of the  $j^{th}$  best thing happening. In this case, weights (pseudo-probability) are assigned not to a particular state of nature, but to a preference order of the utility. Thus,  $w_1$  is the weight assigned to the best utility and  $w_n$  is assigned to a worst utility. Here, a question that naturally arises is how the agent assigns the weights it is going to use in solving some problem. At the fundamental level, the answer is that a human expert interacting with the agent subjectively assigns it. But this may be a hard job in autonomous environments. Thus, we propose a method to assign the weight vector automatically reflecting *degree of optimism* of agents. Using OWA operator, the degree of optimism is defined below [14].

$$Opt(W) = \sum_j w_j (n - j) / (n - 1)$$

Using this definition, we propose the method to assign weight vector automatically. Users of agents subjectively decide upon their degree of optimism  $Opt(W)$ . They then input this value into a following linear programming equation

$$\max - \sum_j w_j \log_2 w_j$$

Subject to:

$$Opt(W) = \sum_j w_j (n - j) / (n - 1)$$

$$Opt(W) \in [0, 1]$$

$$\sum_j w_j = 1$$

$$w_j \geq 0 \text{ for } j = 1, 2, \dots, n$$

This approach is closely related to the maximum entropy method used in probability theory, which is a commonly used rationale for selecting a canonical probability distribution from among a set of relevant ones.

Advantage of this method is that for various cardinalities of OWA, we can consistently provide weights corresponding to given  $Opt(W)$ . Using this method, we can treat decision making under uncertainty (problem of *Level-3*) within the previously mentioned framework of decision making under risk (problem of

Level-2) since we can view OWA operator as pseudo-probability distribution. In this paper, from now on, we regard Level-2 and Level-3 as the same problems.

#### 5.4. Analyzing opponents' moves

Using decision making method mentioned above, the agent can decide upon optimal strategy even under uncertainty. However, in order to get a stable utility, the agent should reduce uncertainty by analyzing opponents' moves and updating its belief. In order to model belief updating process, we devise the game theoretic analyzing method using *dynamic belief network* (DBN). DBN provides a mechanism to foresee the probability of interest in the next state with regard to the current beliefs. That mechanism is called probabilistic projection and can be performed by a three step updating cycle called roll-up, estimation, and prediction phases which avoid the structure of the network growing indefinitely, as suggested in the dynamic model. Specifically, the reasons that support using DBN are:

- (a) The cause-effect relationship between concepts can be concise and intuitively represented by the links between the nodes in the network;
- (b) The uncertainties can be managed by the conditional probabilistic table between linked nodes;
- (c) The temporal changes may be properly handled with DBN.

There are two kinds of models in our DBN, *sensor model* and *state evolution model*. The role of sensor model is to obtain information related to other agents' strategies and/or behaviours (such as signal), and next state is estimated by state evolution model based upon agents' action and prediction of current state. Specifically, we consider following values of states.

- Type of agent (T):  
In order to predict other agents' strategies, agents have to know other agents' preference relation. Here, *type* is a value which decides each agent's preference relation. Specifically, *type* represents each agent's degree of risk aversion, as mentioned above. For instance, for an agent having three types (Risk prone, Risk neutral and Risk aversion), its type value  $T$  is represented by  $T = \{-1, 0, 1\}$ .
- Knowledge Hierarchy Structure (K):  
For the strategic decision making, agents have to analyze not only other agents' types but also *knowledge hierarchy structure* [10]. Knowledge hierarchy structure is the hierarchical structure of each agent's knowledge such as:  
"whether or not all agents know all agents types,"

"whether or not all agents know 'all agents know all agents types'," ..., etc.

For instance, in case of competition among two agents (*agent 1* versus *agent 2*), the knowledge hierarchy structure is defined as follows.

1. Whether or not *agent 1* knows type of *agent 2*
2. Whether or not *agent 2* knows type of *agent 1*
3. Whether or not *agent 1* knows "whether or not *agent 2* knows type of *agent 1* own"
4. Whether or not *agent 2* knows "whether or not *agent 1* knows type of *agent 2* own"

We can represent knowledge hierarchy structure of this example into four states ( $\theta_1 \sim \theta_4$ ) as shown in Table 1. In case of state  $\theta_2$ , it is said that types of all agent is *Common Knowledge*.

**Table 1. Knowledge hierarchy structure**

Agent 2 knows Agent 1	No	Yes	Yes	No
Agent 1 knows Agent 2	Yes	Yes	No	No
State	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$

Note that although four states exist, information partitions of each agent ( $P_1$  and  $P_2$ ) are

$$P_1 = \{(\theta_1, \theta_2), (\theta_3, \theta_4)\} \text{ and } P_2 = \{(\theta_1, \theta_4), (\theta_2, \theta_3)\}.$$

Thus, all we have to do is to compute only two states within information partition of each agent.

- Belief of Belief (B):  
With analyzing knowledge hierarchy structure mentioned above, we can analyze whether or not the opponent knows my type exactly, etc. Although it is better to analyze "if opponents don't know my type exactly, how do opponents misunderstand my type?" Here, *belief of belief* is a concept such as "I believe opponent believes my type is  $x$ ". For example, if agent 1 has three types, its belief of belief is represented by  $B = \{-1, 0, 1\}$ .
- State of Nature (N):  
State of nature is strategies of the other agents as mentioned above. For example in case of competition between two agents, if the opponent has three strategies, it is represented by  $N = \{\omega_1, \omega_2, \omega_3\}$ .
- Existence of Signals (I):  
Whether or not, agent can get signals (information) about other agents' strategies. Simply, it is represented by a Boolean variable. Namely,  $I = \{T, F\}$ .  
Considering these elements, we model sensor and state evolution model. Example of the *sensor model*

using two-slice temporal belief network (2DBN) is shown in Fig. 4.

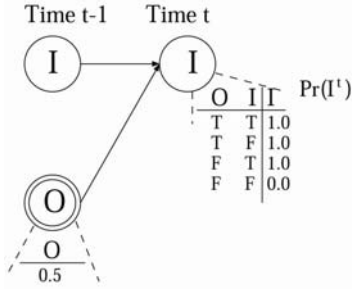


Figure 4. Example of sensor model

Agents get and analyze the signal using sensor model. But agents cannot always get signals successfully. Therefore, we add event  $O$  into the sensor model.  $O$  is an event which shows agents obtain signals. And by adding conditional probability table (CPT), we can construct the sensor model. In this example, we set that agents can get signals with probability 0.5.

Generally, we can analyze probability of getting signals  $Pr(O|S^{t-1})$  using the sensor model. Example of the state evolution model represented by 2DBN is shown in Fig. 5.

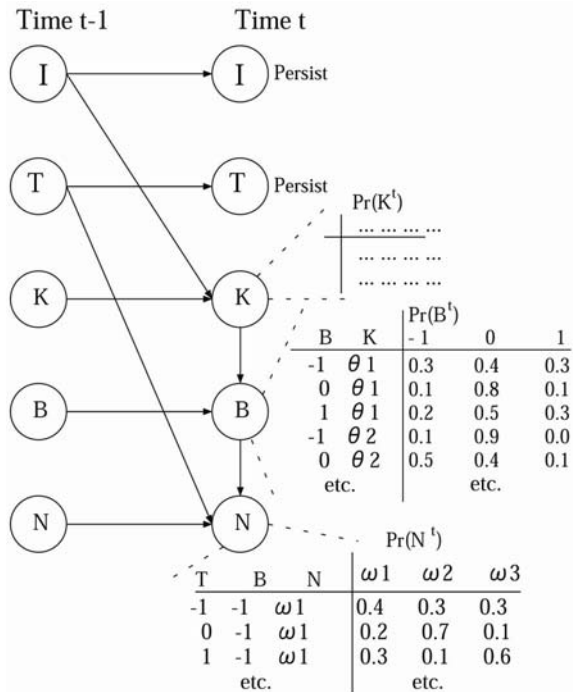


Figure 5. Example of state transition model

It represents the effects of adoption of some strategies as direction of link. In the case of agent competition, we want to know the probability

distribution over states of nature  $N^t$ . As shown in Fig. 5, value of  $N^t$  requires knowing  $T^{t-1}$  and  $N^{t-1}$ . Moreover,  $N^t$  and  $B^t$  are correlated and  $B^t$  and  $K^t$  are also correlated. Relations and correlations of each state is described via conditional probability form since each state values includes uncertainty. Thus considering relations of each state, probability distribution of current state of nature is calculated as follows.

$$\Pr(K^t, B^t, N^t | S^{t-1}) = \Pr(N^t | B^t, S^{t-1}) \times \Pr(B^t | K^t, S^{t-1}) \times \Pr(K^t | S^{t-1})$$

where  $S^{t-1}$  is a previous state.

And finally, by multiplying probability of getting signals calculated by sensor model and probability of current state of nature obtained by state evolution model, agent can update its belief and reason about the opponents' strategies.

### 5.5. Example

Competition model presented in Section 5 is applied to competition among dealer agents, in the electronic commerce project [4] for a typical contracting scenario. In this scenario, a customer agent negotiates with a number of dealer agents, and each dealer agent tries to identify customer's needs and takes parts in bidding to win the contract. Risk and uncertainty are integral parts of business and are especially important in this scenario.

In order to win the contract, dealer agents must not bid too high because it will fail to get the contract and loses the time and money spent in preparing the proposal. On the other hand, if the dealer agent bids much lower than its competitors, it loses again because, it obtains the contract but it has undertaken to fulfill it at a price far lower than necessary. Therefore, to win a contract, the dealer agent must bid high enough to make a profit but low enough to get a job both at the same time. In order to do so, strategic prediction and analysis of competitors' movement is indispensable [10]. The model proposed in Section 5.3 can cope with this problem.

In this example, we assume that a customer agent negotiate with two dealer agents. Both dealer agents are selling similar kinds of products, means that there is no product differentiation. This is a simplified model but expressive enough to explain the ideas. Thus inevitably, the dealers will fall into the price competition at limitation on price.

Here, we assume each dealer agents has two strategies, to sell the products with "High Price (\$40,000)" and "Low Price (\$30,000)". We assume that \$30,000 is a limitation on price for both dealer agents. Payoff matrix of this price competition is shown in

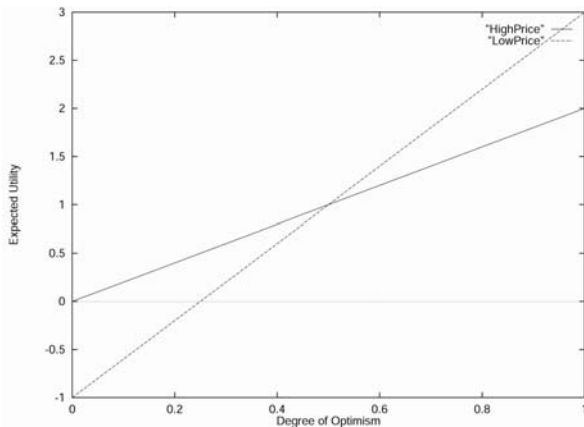
Table 2. This game environment is analyzed as *Chicken Game* in micro economics theory.

**Table 2. Payoff matrix of price competition**

	High price	Low price
High price	(2, 2)	(0, 3)
Low price	(3, 0)	(-1, -1)

In case that, both dealer agents adopt “High Price” strategy, possibility to win the contract is half because of no product differentiation, means that both agent’s payoff is 2. In case that dealer agent 1 adopts “Low Price” and opponent adopt “High Price”, dealer agent 1 can win the contract and obtains payoff 3 and vice versa. In case that, both agent adopt “Low Price”, since \$30,000 is limitation on price, both agent’s benefit will be deficit because, although they are selling products at low price as much as possible, possibility to win the contract is very low (at most half). In this case, we assume that both payoffs are -1.

In this game environment, best response strategy is to adopt the different strategy from their opponents, (Low Price, High Price) and (High Price, Low Price). At the first glance, it seems that this game is easy to solve, but since opponents’ strategy cannot be predicted completely, situation gets quite complicated. To cope with this complexity derived from uncertainty, we apply our competition model. First, we apply decision making model based on degree of optimism introduced in Section 5. For instance, if  $Opt(W)=0.7$ ,  $[w_1, w_2] = [0.7, 0.3]$ . Thus for “High Price”, expected utility is  $0.7 \times 2 + 0.3 \times 0 = 1.4$  and in case of “Low Price”,  $0.7 \times 3 + 0.3 \times (-1) = 1.8$ . Thus the agent should adopt “Low Price”. Expected utility with changing degree of optimism is shown in Fig. 6.

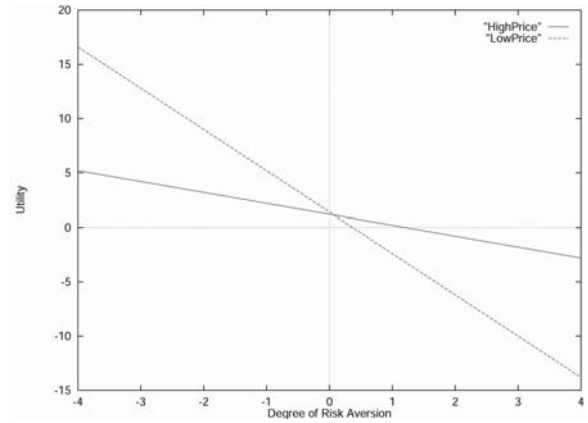


**Figure 6. Expected utility with degree optimism**

As it is shown, until  $Opt(W)<0.5$ , dealer agent adopts “High Price” and when  $Opt(W)>0.5$ , then “Low Price” is adopted. This is because, since “Low Price” has an aspect of high risk high return, in case that dealer

agent is optimistic ( $Opt(W)> 0.5$ ), it prefers “Low Price” but when it becomes pessimistic ( $Opt(W)< 0.5$ ), it prefers “High Price” which has a character of low risk low return. This experimental result proves that using the proposed model, interactive decision making is possible even though probability distribution over moves of competitors is completely unknown.

Next example is related to the risk management method presented above. We assume that, dealer agent 1 believes opponent adopts “High Price” with probability of 0.6 and “Low Price” with 0.4. Degree of utility with changing degree of risk aversion ( $\eta$ ) is shown in Fig. 7.



**Figure 7. Expected utility with degree of risk aversion**

As it is shown, when dealer agents’ preference is risk aversion ( $\eta > 0$ ), it prefers “High Price” strategy which has a character of low risk low return, because utility of “High Price” is higher than that of “Low Price”. And in case that dealer agents’ preference becomes risk prone ( $\eta \leq 0$ ), high risk high return strategy, i.e., “Low Price”, is adopted. This experimental example proves that our risk management model allows agents make decision with reflection of users’ opinion within a simple representation.

There are several advantages to use competition model presented in Section 5.3. First, we can analyze opponents’ private knowledge. It is quite important to deal with uncertainty and hostility under a society of heterogeneous agents. This is because, since each agent has a different preference, ability to compete, belief etc., each must have a model of other agents’ private knowledge in order to predict other agents’ future moves. Proposed model makes it possible by introducing the concept of knowledge hierarchy structure and belief of belief. Second, because of introduction of degree of optimism, agent can decide competitive strategy even if probability of competitors’ movement is completely unknown. From the viewpoint of behavioural psychology, utilization of degree of optimism is natural and from engineering point of view is practical.

## 6. Conclusion

In this paper, we addressed the quality and complexity issues of MAS and proposed metrics to measure the complexity. The metrics were used for devising a candidate set of agents for MAS design. We also addressed the knowledgeability and uncertainty of MAS by devising a model and technique to cope with uncertainty in competitive situations.

## References

- [1] Albrecht, A.J. and Gaffney, J.F., Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation, IEEE Trans. Software Engineering, vol. 9, no. 6, pp. 639-648, 1983.
- [2] Boehm, B., Software Risk Management, IEEE Computer Society Press, CA, 1989.
- [3] Crosby, P.B., Quality is Free: The Art of Making Quality Certain, McGraw-Hill, New York, 1988.
- [4] Far, B.H. et al, An Integrated Reasoning and Learning Environment for WWW Based Software Agents for Electronic Commerce, Transactions of IEICE, vol. E81-D no. 12, pp. 1374-1386, 1998.
- [5] Far, B.H., Agent-SE: A Methodology for Agent Oriented Software Engineering, Enabling Society with Information Technology, Q. Jin et al. eds., pp. 357-366, Springer, 2001.
- [6] Onjo, H., and Far, B.H., A Unified View of Heterogeneous Agents' Interaction, Transactions of the IEICE, vol. E84-D, no. 8, pp. 945-956, 2001.
- [7] Ichikawa, A., Decision Making Theory, Kouritsu Pub., 1983. (in Japanese).
- [8] Juran, J.M., Gryna, F.M. Jr., Bingham, F.M. (eds.), Quality Control Handbook (3<sup>rd</sup> edition), McGraw Hill, New York, 1979.
- [9] Kan, S.H., Metrics and Models in Software Quality Engineering, Addison-Wesley, 1995.
- [10] Kajii, A. and Matsui, A., Micro Economics: A Strategic Approach, Nihon-Hyouron Pub., 2000. (in Japanese).
- [11] McCabe, T.J., A Complexity Measure, IEEE Transactions on Software Engineering, vol.2, no.4, pp. 308-320, 1976.
- [12] McCall, J.A., Richards, P.K., Walters, G.F., Factors in Software Quality, RADC TR-77-369, 1977. Vols I,II,III', US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.
- [13] Sowa, J.F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, 1984.
- [14] Yager, R.R., On Ordered Weighted Averaging Aggregation Operators in Multi-Criteria Decision Making, IEEE Trans. SMC, no. 18, pp. 183-190, 1988.
- [15] Yager, R.R., Decision Making Under Dempster-Shafer Uncertainties, International Journals of General Systems, vol. 20, pp. 233-255, 1990.