

# Knowledge-based Software Engineering

Proceedings of the Fifth Joint Conference on  
Knowledge-based Software Engineering

Edited by

Tatjana Welzer

*Faculty of Electrical Engineering and Computer Science,  
University of Maribor, Maribor, Slovenia*

Shuichiro Yamamoto

*NTT Data Corporation, Tokyo, Japan*

and

Ivan Rozman

*Faculty of Electrical Engineering and Computer Science,  
University of Maribor, Maribor, Slovenia*



Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

# Software Agents for Uncertain and Complex Environments

Behrouz HOMAYOUN FAR  
Faculty of Engineering, University of Calgary  
far@enel.ucalgary.ca

**Abstract.** Complexity, knowledgeability and uncertainty issues of multi-agent systems (MAS) are discussed. Complexity of MAS is composed of structural and algorithmic complexity. We define metrics to measure the complexity of MAS. The metrics are used for devising a candidate set of agents for MAS design. Knowledgeability of MAS is defined in terms of problem solving and cognitive capabilities and the ability to cope with agents' interactions, such as cooperation, coordination and competition. Uncertainty usually arises when agents are engaged in competitive tasks. We 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.

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 software 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 [9], Boehm's [2], McCall's [12], and ISO 9126 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. 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 is either structural or algorithmic. 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.

## 2. 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. Complexity of MAS is both *structural* and *algorithmic*. They both can be defined in either objective or subjective way.

### 2.1 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. Two kinds of dynamic relationships can be devised: *interactions* among subsystems and *intra-actions* within subsystems [5]. 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.

## 2.2 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.

## 2.3 MAS Complexity 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.

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.

## 2.4 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  $UF_eC$ .

## 3. 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. Knowledgeability can be defined in terms of:

- *Interactions*, i.e., the ability to directly communicate or collect data on the other agents.
- *Cognitive capabilities*, i.e., the ability to manipulate knowledge and make decisions based on the knowledge and collected data.

The following subsections elaborate this.

### 3.1 Cognitive Capabilities

Three main capabilities of agents in MAS are *representing*, *using* and *sharing* the knowledge. A modified version of semantic net called *Symbol Structure (SS)* is used to represent individual agent's knowledge structure [5]. Ability to use the knowledge is realized by having the knowledge-base in the form of SS and mechanisms for problem solving using the SS [5]. Finally, ability to share the knowledge depends on ontologies for the domain and task. Mechanisms for using and sharing are presented in [6].

### 3.2 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.

### 3.3 Decision Making Mechanism in MAS

Figure 1 shows decision making mechanism based on agents' interaction. Agents engaged in cooperative and coordinative tasks usually have precise information about the other agents' goals due to the fact that direct communication between the agents is possible. Therefore the predictive model is usually deterministic. In case of competition, the agent must predict the other agents' goals based on the signals from the opponents rather than direct communication. Thus the predictive model is usually non-deterministic.

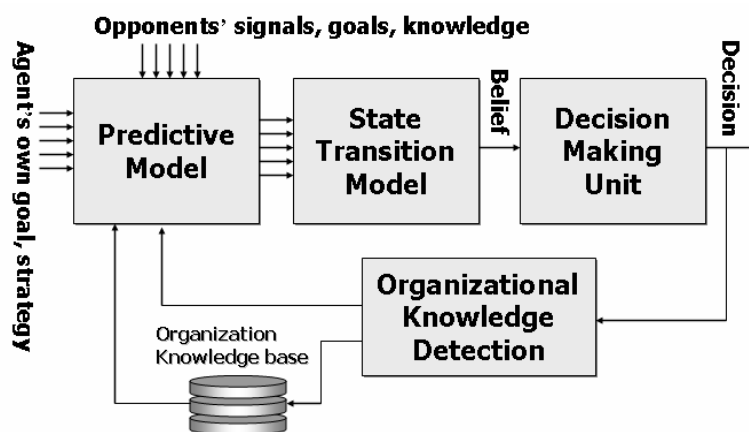


Figure 1. Decision making mechanism

## 4. Uncertainty in MAS

In this section, we consider multi-agent interaction under competitive and uncertain environments. Information gained through agents' behaviour, i.e., *signals* may be incomplete. 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.

#### 4.1 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.

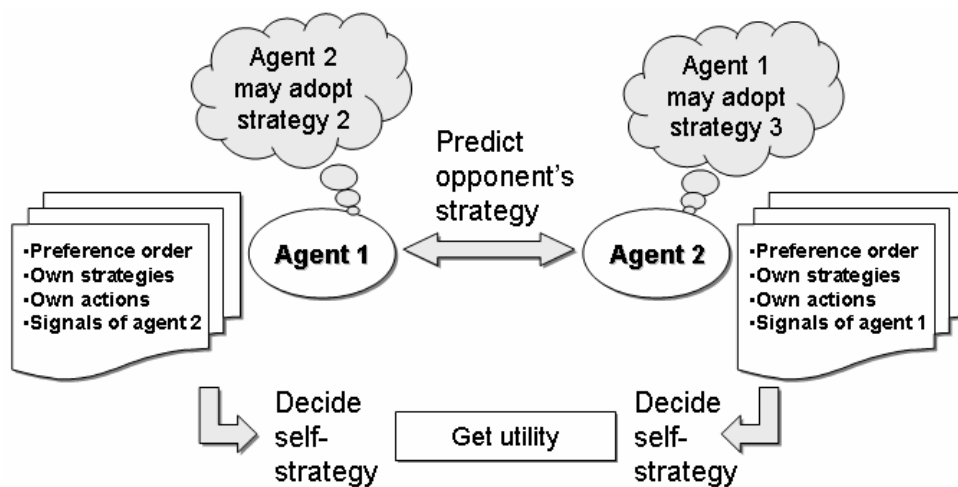


Figure 2. Overview of agents' competition

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' moves is *uncertain* because they may try to hide their strategies and the knowledge behind it as much as possible. The work presented here suggests modeling an incomplete game theoretical based decision making method for competitive agents.

#### 4.2 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.

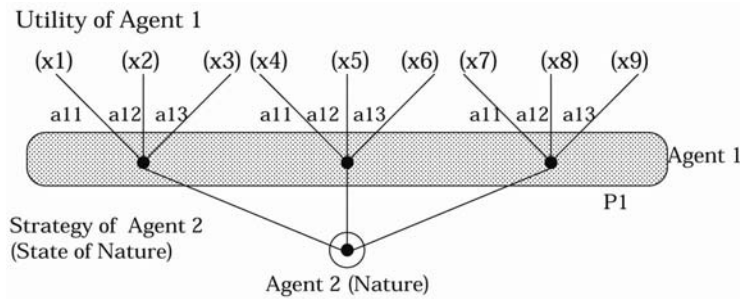
As for first, we treat them based on normal game theoretic approach. As for second, we regard these agents as *natural* objects and their strategies as *state of nature* and treat them

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*).

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* 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 Bayesian perfect Nash Equilibria [10].



**Figure 3. Example of competition model**

A question which naturally arises here is how each agent assigns his belief autonomously. The answer can be achieved by dividing uncertainty into levels. 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, decision making is straightforward. The agent selects the strategy which maximizes its utility using traditional game theory.
- **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*).

#### 4.3 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 adopt 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.

#### 4.4 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 [13]. 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  $j^{\text{th}}$  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 [14]. In particular we can view  $w_j$  as a kind of probability that of the  $j^{\text{th}}$  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 [13].

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

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:

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

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

$$\sum_j w_j = 1$$

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

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.

#### 4.5 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. A method for belief update using *dynamic belief network* (DBN) is presented in [6].

## 5. Conclusion

In this paper, we addressed the complexity, knowledgeability and uncertainty issues of multi-agent systems (MAS) and defined metrics to measure the complexity of MAS. We also defined knowledgeability of MAS in terms of problem solving and cognitive capabilities and the ability to cope with agents' interactions. Finally, we devised models and techniques 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, RADDC TR-77-369, 1977. Vols I,II,III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.
- [13] Yager, R.R., On Ordered Weighted Averaging Aggregation Operators in Multi-Criteria Decision Making, IEEE Trans. SMC, no. 18, pp. 183-190, 1988.
- [14] Yager, R.R., Decision Making Under Dempster-Shafer Uncertainties, International Journals of General Systems, vol. 20, pp. 233-255, 1990.