PAPER *Special Issue on Information and Systems for the IEICE Transactions*

Goal-oriented Methodology for Agent System Development

Zhiqi SHEN[†a)], **Chunyan MIAO**[††], ***Members***, **Robert GAY**[†], ***and*** **Dongtao LI**[††], ***Nonmembers***

SUMMARY The Goal-Orientation is one of the key features in agent systems. This paper proposes a new methodology for multi-agent system development based on Goal Net model. The methodology covers the whole life cycle of the agent system development, from requirement analysis, architecture design, detailed design to implementation. A Multi-Agent Development Environment (MADE) that facilitates the design and implementation of agent systems is presented. A case study on an agent-based e-learning system developed using the proposed methodology is illustrated in this paper.

key words: *Goal-oriented Modeling, Agent-oriented Software Engineering, Goal Net*

1. Introduction

Agent system represents a new paradigm in software engineering. The autonomous, cooperative, intelligent and goal-oriented characteristics of agent make agent system a promising solution for next generation of software in various application domains [1].

Agent system is increasingly attracting industrial interests. Meanwhile, agent-oriented software engineering has become an active research area since late 1990s. Some development methodologies such as Gaia [3], [6], Tropos [7], etc., and agent development tools like JADE [8] etc., have emerged. However, there is still a lack of widespread of development and deployment of agent systems. The major reason is that the research on narrowing the gap between agent mental model and agent implementation is rare [3]. The transition from agent design to agent implementation is an important step in defining a complete agent software process [12].

To fill up the gap between agent model, design and implementation, we propose a comprehensive methodology for agent system development. This methodology is based on Goal Net [2], [13], a goal-oriented model for modeling agent's goals and managing the mental states of an agent. A Multi-Agent Development Environment (MADE) that facilitates the design and implementation of agent systems has been developed. It has been integrated with JADE, which is a FIPA compliant agent construction tool, to provide a complete agent development environment.

Manuscript received January 1, 2003.
Manuscript revised January 1, 2003.
Final manuscript received January 1, 2003.

†The author is with the School of EEE, Nanyang Technological University, Singapore
††The author is with the School of Computer Engineering, Nanyang Technological University, Singapore
a) E-mail: zqshen@ntu.edu.sg

Goal is a desired state that an agent intends to reach. Goal Net is a composite goal hierarchy which is composed of *goals* and *transitions*. The *goals*, represented by circles, are used to represent the goals that an agent needs to go through in order to achieve its final goal. The *transitions*, represented by arcs and vertical bars or rectangles, connect one goal to anther specifying the relationship between goals it joins. Each transition must have at least one *input goal* and one *output goal*. Each transition is associated with a task list which defines the possible tasks that agent may need to perform in order to transit from the input goal to the output goal. Figure 2 shows a simple goal net.

There are two types of goals in a Goal Net model, *atomic goal* and *composite goal*. An *atomic goal*, represented by a blank circle, accommodates a single goal which could not be split anymore; a *composite goal*, represented by a shadowed circle, may be split into goals (either composite or atomic) connected via transitions.

In Goal Net models, there are four types of basic temporal relationships between goals: *sequence*, *concurrency*, *choice* and *synchronization*. *Sequence* relation represents a direct sequential relationship between one input goal and one output goal; *concurrency* relation has one input goal but more than one output goals, and all its output goals can achieve simultaneously; *choice* relation specifies a selective connection from one goal to other goals; *synchronization* relation specifies a synchronization point from different input goals to a single output goal. With different combinations of the basic temporal relations, Goal Net supports a wide range of complicated temporal relationships among goals. This is one of the major differences between Goal Net and other goal modeling methods.

With such a composite goal hierarchy and various temporal relationships within the hierarchy, a complex system can be recursively decomposed into sub-goals and sub-goal nets. In such a manner, the system can be easily modeled and simplified.

The proposed goal-oriented (GO) methodology covers four phases, namely Requirement Analysis, Agent Architecture Design, Detailed Design and System Implementation.

Requirement Analysis: In this phase, system requirements are modeled as different goals which need to be achieved. The given problem is modeled in the manner that what goals need to be achieved, what are the possible ways to achieve these goals and what are the relations among different goals. The objective of this phase is to produce a preliminary high level goal net.

Agent Architecture Design: The preliminary goal net is converted to an agent hierarchy by splitting the goal net using the goal-split policies. Agent communication protocols and system architecture are also defined in this phase.

Detailed Design: In this phase, for each identified agent, goals and sub-goals are refined to be closely bound to the agent. Tasks, perception, knowledge and message handler for each agent are also specified in details.

System Implementation: In the implementation phase, the detailed design specifications in the previous phases are used for agent implementation. The detailed implementation includes task development, task selection mechanism, goal selection mechanism, knowledge management and agent implementation.

Following this introduction, Sect. 2 describes an e-learning case study used in this paper. The details of the four phases are illustrated in Sect. 3 with examples from the case study. Section 4 gives an evaluation of the proposed methodology against other agent methodologies from the perspective of software engineering. The conclusion is reached in Sect. 5.

2. E-Learning Case Study

In this paper, an e-learning case study is used to illustrate the proposed methodology. An agent-based e-learning system developed using the proposed GO methodology is presented.

The case study is elicited based on a small IT enterprise, who provides customized e-learning services in the e-learning grid to some small and medium size enterprises that belong to an Electronic Business Network (EBN). In order to train some developers Java programming language, a member company in EBN has decided to adopt online e-learning services. Currently there are three related service providers in the e-learning grid. These service providers offer different courses at different prices for different technical levels, and the courses are delivered in the form of learning objects (LOs). Tests are also provided by the services for the assessments of learning progress.

To help the employees acquire the new knowledge easily, an agent based system has been designed to provide a learner centric e-learning system. The system provides the following functions:

- Automatically generate a learning path for an employee based on his/her learning goal and current skill level;
- Select and deliver learning courses from different providers to learners;
- Provide personalized help to the learner in the self-learning cycle. The self-learning cycle should be repeated until the results of the learning assessment meet the requirements.
- Evaluate and minimize the cost and learning duration for learners.

In the following sections, we will stick on this case study to illustrate how to use the proposed methodology to develop a multi-agent e-learning system.

3. Development Phases

In this section, the four development phases of the proposed methodology are explained in details. To make things clearer, the e-learning case study is used to illustrate the methodology step by step.

3.1 Requirement Analysis

Requirement analysis is the initial phase in many software development methodologies. In the proposed methodology, Goal Net serves as a problem modeling and analysis tool from the beginning of the requirement analysis.

Goals are seen to have substantial promise in aiding the elicitations and elaborations of requirements. For example, KAoS methodology [9] uses goal as the key concept in requirement acquisition. Anton [10], [11] also uses goal as the main guiding concept in developing requirement specifications. In the requirement analysis phase, the objective is to derive a preliminary goal net by identifying goals (*what*), possible tasks for achieving the goals (*how*) and the environment that may affect how goals are pursued (*situation*). Unlike the existing goal-based methods, which analyze an agent's goal isolated from its environment, Goal Net analyzes an agent's goal together with the agent's dynamic environment. It supports goal selection and action selection mechanisms for pursuing the agent's goal in a dynamic environment.

The first task in this phase is to identify all the goals and construct a goal hierarchy without transitions. Following a top-down approach, the analysis starts from the overall goal (root goal in a goal net) to solve a complex problem. This goal is decomposed into a set of sub-goals to solve each decomposed problem. Each sub-goal can be further decomposed to a set of sub-goals. Such kind of goal decomposition continues until all the goals in the goal hierarchy can be easily achieved or can be solved through its sub-goals.

In the e-learning case, the root goal, which is the final goal of the system, is to provide personalized Java learning to employees. This goal can be directly decomposed into three sub-goals, *Generate learning path*,

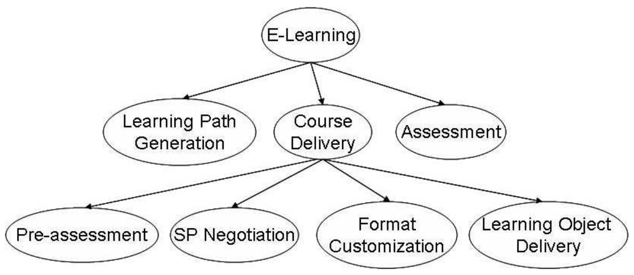


Fig. 1 High level goals in our e-learning system.

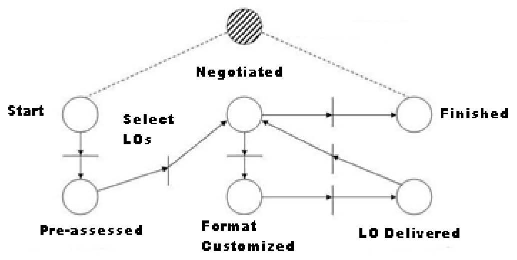


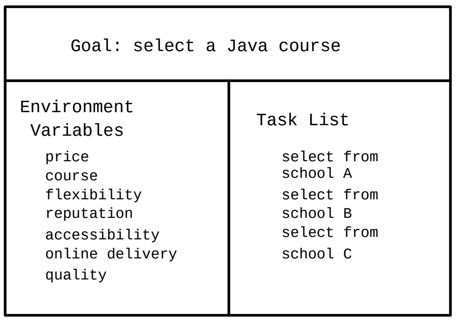
Fig. 2 Goal net for course delivery.

Learning object delivery and *Assessment* as shown in Fig. 1. However, each of these three goals is still a complex problem to solve. We can further decompose these goals. For example, *Course Delivery* can be decomposed to *Pre-Assessment*, which conducts an assessment on learner to decide whether the learner is eligible to skip a specific learning object, Service Negotiation, which is to negotiate the details of the course delivery; *Format Customization*, which is to customize the standard course format into a user preferred format, and *Course Delivery*, which takes care of the real course transition and presentation. In such a manner, a goal hierarchy is constructed.

The next step is to identify the interactions or relationships among goals in the goal hierarchy, that is, add transitions into the goal hierarchy. Without considering the details of the tasks associated with each transition, all the possible connections among goals and corresponding tasks should be figured out based on different temporal relationships. The result after this step is a preliminary goal net.

Figure 2 illustrates how to identify transitions to form a goal net. As shown in Fig. 1, *Course Delivery* goal is decomposed into four sub-goals. To form a goal net, we connect all the goals which may possibly have direct interaction between each other. This is trying to map the entire scenarios into the goal net.

The last step in the requirement analysis phase is to model the environment. The general working environment of agents may include physical environments such as computers and printers; software environment such as operating systems, communication protocols and database; and application specific environment such as application domain. In the requirement analysis phase, we only concentrate on the application specific environment because the rest two can be decided in the detailed design. As different applications use different methods to model its environment, it is difficult to provide a general model for environment modeling. However, a common method is to use variables to represent the environment sources and the values of the variables indicate the environment changes.



Goal: select a Java course	
Environment Variables price course flexibility reputation accessibility online delivery quality	Task List select from school A select from school B select from school C

Fig. 3 A simple GET card.

To simplify the process to identify goals, transitions and the environment, *Goal-Environment-Task* (GET) card is created. Each GET card contains a goal, the possible tasks to achieve the goal, and the environment variables. These three elements capture the essential dimensions of goal modeling based on Goal Net. It can be created by designers or even customers who are not familiar with Goal Net. Figure 3 shows a simple GET card illustrating the goal of selecting a java course from different schools.

The result of the requirement analysis is a complete preliminary goal net, within which the goals, transitions, the environment variables and the possible task lists associated with each transition, are clearly defined.

3.2 Agent Architecture Design

Given a complex goal net with hundreds of goals, it is very difficult to create a single agent pursuing all the goals. Instead, the goal net can be split into a set of sub-goal nets, and these sub-goal nets are used as the goal models for different agents. As a result, a multi-agent system can be derived from the goal net. Agents in the multi-agent system modeled by Goal Net are organized in a hierarchy structure, called *Agent Hierarchy.* A higher level agent becomes the coordinator of the lower level agents. At the same time, it also pursues its own goals. In such a manner, the root goal in the original goal net becomes the common goal of the derived multi-agent system. The transitions between goals will be used for the agent coordination and synchronization.

In order to derive a multi-agent system from a goal net, we need to create agent identification policies and split the goal net according to these policies. There are several factors affecting the agent identification policies:

- *Modularity*: each split goal net solves a subproblem.
- *Reusability*: each split goal net can be reused to take part in a new composition for solving a particular problem.
- *Location*: each split goal net is for a distributed agent.
- *Load-balancing*: a goal net is split so that agents have balanced work load.
- *Organizational role*: each split goal net is corresponding to a role in an organization.

The agent identification policy can be designed according to the system requirements. Policies dealing with different factors can be applied to the same goal net split. For example, at higher level split, *Location*, *Load-balancing* or *Modularity* can be used whereas at lower level split, *Organizational role*, *Reusability* or *Modularity* can be used accordingly.

In Goal Net model, each goal is allowed to have at most one parent goal; similarly in the agent hierarchy each agent can only have one coordinator agent. The coordinator agent is responsibility for coordinating the child agents according to the original goal net. The original goal net becomes the coordination plan of the generated multi-agent system.

In the e-learning case, we apply *Modularity* agent identification policy to construct the agent hierarchy. We simply take the root goal as the e-learning coordinator agent, and create three different agents pursuing *Learning Path Generation* goal, *Course Delivery* goal and *Assessment* goal respectively. *Learning Path Generation Agent* is to dynamically generate the learning path for a learner; *Course Delivery Agent* is responsible for delivering the correct course content to the learner in an appropriate format; *Assessment Agent* provides the assessment of each course the learner has taken and evaluates the test results. *E-learning Agent* is a coordinator for these three agents. It is still reasonable to create more agents to pursue lower level sub-goals. For example, we can create four separate agents to pursue *Course Delivery* goal according to Fig. 1 and the *Course Delivery Agent* will be their coordinator.

After the agent hierarchy has been derived, the next step is to design the communication protocols for agents' interaction. The protocols designed at this stage are conceptual as the real protocols rely on the technology used and the agent running platform, which will be decided in detailed design phase. The conceptual protocols define the messages required for agent communication. The format of the message is application independent. A message consists of the following attributes:

- *Message ID*: a unique identifier for each message.
- *Message Source*: the identifier of the message sending agent.
- *Message Destination*: the identifier of the message receiving agent.
- *Message Type*: the type of content in the message.
- *Message Content*: the real content data of the message.

In the above attribute list, *Message Type* and *Message Content* are application dependent. The message receiver handles the message content based on the message type. *Message ID* is automatically generated during agent runtime.

The output of this phase is an agent hierarchy with clear definition of coordination and communication among agents.

3.3 Detailed Design

In the previous two phases, the skeleton of the multi-agent system has been developed. In this phase, the details of the system, such as goal details, task details, agent communication language, environment perception mechanism etc., should be designed. The output of this phase should be able to guide the actual implementation of the agent system, which will happen in the next phase.

In the preliminary goal net, all the goals are conceptual and do not have much details, so the first task in this phase is to fill in the details for each goal. This includes the attributes of each goal, the goal achievement function, the goal selection function and other application specific goal functions. The attributes specify the properties of a goal, and they are in a variable-value manner. The data types of the attributes should be specified in this step to guide the implementation.

After specifying the details of goals, we need to design the details of all the transitions, including the tasks in each transition. Each *task* associated to a transition is independent of that transition, which means different transitions can reuse the same task. All the task functions can be designed in this stage.

The environment is also designed in details. This includes: the environment variables, of which the data types and possible values should be designed; the environment interface, which defines the way that the agent perceives and interacts with the environment; and the environment management, which defines how the environment variables are changed to represent the latest situation.

To make the agent run with social interaction, the details of the agent communication should also be worked out here. The conceptual protocols designed in the last phase should be converted into detailed design according to the agent platform and the real agent communication language. The message handling mechanism for each agent, which defines how an agent deals with different incoming messages under different situations, should be developed.

An agent modeled by Goal Net lives in a PR2A life cycle, Perceiving the environment, Reasoning for its next goal (goal selection), Reasoning for its next action for achieving the selected goal (action selection) and Acting in the environment. The design of goal selection mechanisms and action selection mechanisms are important parts in the detailed design.

3.3.1 Goal Selection

The goal achievement function is to calculate the achievement of the goal pursuit. It is important for the goal measurement. If the partial goal achievement is used in the system, then the threshold value for decision making must be set for each goal. Goal selection function is used to select the next goal based on the goal selection algorithm in Goal Net [2]. The general factors that will affect the goal selection include the current goal achievement, the environment situation, cost, time, system specified constraints etc. The goal selection function makes the agent system goal autonomous.

Goal Net supports different reasoning mechanisms for goal selection based on environment variables, constraints defined in real applications. In the E-learning case, the following environment factors have been taken into account for selecting the next goal: *course price*, *learning duration*, *learner's skill level* associated with a course and *learner's expectation* (such as, minimum costs, shortest duration or highest skill level). The agent will select the next course (goal) through a utility function according to the real values of the above factors. Other reasoning mechanisms such as rule-based reasoning have also been used in the E-learning case. For example, whether the course needs to be re-delivered is based on some rules according to the learner's assessment results. The flexibility to choose different reasoning mechanisms in a goal net is one of the key advantages of Goal Net model.

3.3.2 Action Selection

Task/action selection mechanism helps an agent to select a suitable task from the task list to pursue the goal, and it is the key for the behavior-autonomy of an agent. In this step, the environment variables defined in the GET card, the time, the cost and other factors affecting the task selection should be designed in details. Goal Net supports flexible actions selection mechanism such as rule based, probabilistic, fuzzy mechanisms etc.

In our E-learning case, a learner may use different terminals like laptop, desktop or PDA to access the course content. Different terminals have different computing capability, so we customize the same course content into different formats, either in pure text or with picture and animation, by applying different tasks during format customization based on a rule-based reasoning mechanism. In the transition *select LOs* in Fig. 2, a *Bayesian Network* is formed based on the environment factors for action selection, which is used to decide whether the learner can skip the current learning object or not, according to his/her current knowledge and skills. The details will be elaborated in Sect. 3.5.

Upon this point, the system should be able to be clearly implemented on an agent platform through guidance by the detailed design of this phase.

3.4 Implementation

In the implementation phase, all the detailed design should be mapped to the agent platform. The mapping from design to implementation includes:

- *Goal Net Construction*: This includes constructing the goal net designed through the previous phases and storing it into database. All the information in Goal Net, including goals, goal interactions, transitions and tasks should be stored properly.
- *Task Development*: This includes developing all the tasks in the goal net and meanwhile all the tasks should be complying with the agent platform.
- *Goal / Task Selection Mechanism Development*: This includes implementing all the goal selection mechanisms and task selection mechanisms in the Goal Net.
- *Agent Implementation*: This is the most difficult task in implementation. An agent, which firstly must comply with the agent platform, should also be able to handle how to load and process the goals in database, how to fire the transition by selecting and executing the suitable tasks, how to handle the messages and how to perceive and interact with the environment.

To assist the mapping from design to implementation, the *Multi-Agent Development Environment* (MADE) has been developed. JADE [8] has been integrated with MADE. The advantages of the integration is that by integrating with JADE, MADE is compliant with the standard FIPA [14]; it is able to provide agent communication mechanism that supports the standard agent communication language (ACL) and provide agent management facilities via JADE. With the integration, we obtained both standard compliant agent platform support and goal-oriented intelligent agent development support. MADE is able to assist developers in both design and implementation phases. Figure 4 shows the architecture of MADE.

MADE is developed in Java and it has four major components: *Agent Development Framework*, *Goal Net Designer*, *Goal Net Loader* and *Agent Creator*. An agent implementation consists of three simple steps, 1) design/draw the goal net of an agent using Goal Net Designer, 2) create an agent using Agent Creator and 3)

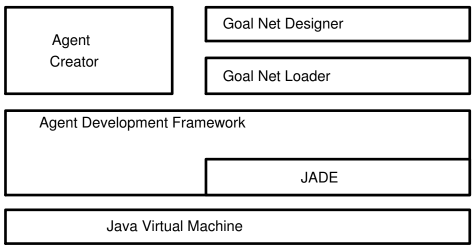
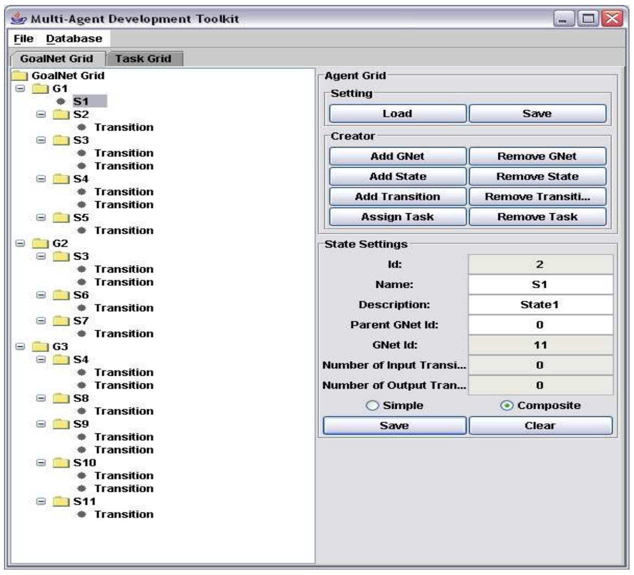


Fig. 4 The architecture of MADE.



Id:	2
Name:	S1
Description:	State1
Parent GNet Id:	0
GNet Id:	11
Number of Input Transi...	0
Number of Output Tran...	0

Simple [✓ Composite]

Fig. 5 Goal Net designer.

load the goal net into the agent using Goal Net loader.

Goal Net Designer is a visualized design tool to help developers in the detailed design of a goal net efficiently. Goal Net Designer provides GUI that allows users to create goals and transitions. The properties of goals and transitions such as attributes, functions, tasks, and action selection mechanisms can be defined.

Figure 5 shows a snapshot of the Goal Net Designer interface. The left panel allows a developer to change the structure of the goal net, and the right panel allows the developer to edit the property of goals, transitions and tasks. Its graphical interface allows developers to edit the design easily. The designed goal net can be automatically saved into predefined database.

Agent Creator is a toolkit helping in creating Goal Net enabled agents. It creates a dummy agent, which is called *Goal Autonomous Agent*, load a goal net from the database into the created agent through the *Goal Net Loader*. Such abilities of the *Agent Creator* and *Goal Net Designer* release developers from the tedious goal management and agent design work, so that they can focus on Goal Net logic and agent tasks development. The components of the agent that *Agent Creator* creates include:

- *Knowledge unit*: provides interface to bind other intelligent engine to process the knowledge data for agent usage.
- *Perception unit*: perceives environment changes.
- *Data unit*: loads/stores data from/into database for the agent to process.
- *Process unit*: processes the Goal Net into agent running order.
- *Compute unit*: implements the task selection and goal selection mechanisms.
- *Action unit*: executes the selected agent tasks.
- *Communication unit*: handles the messages received and messages to be sent.
- *Control unit*: manages the goal pursuit and action execution, and coordinates different components of the agents such as process, communication, perception and action units. Under normal circumstances the control unit executes functions to achieve goals based on the Goal Net model. It obtains the Goal Net model information from the process unit, uses the compute unit to perform goal selection and action selection and executes the actions using the action unit. It instructs the reaction unit to perform reactive action to changes in the environment notified by the perception unit. It decides the order of actions the agent needs to perform. For example, the activities of an agent may include actions for goal pursuit, goal and action selection, environment change reaction, and communications with users or other agents, etc. The control unit schedules the above activities.

Agents created by MADE can work on different duties if different goal net is loaded into its ”brain”. For the E-learning case, with MADE the major work becomes the design of the goal nets for the agents, the implementation of tasks and the action selection mechanisms. To run the system, the dummy agents are created by Agent Creator and different goal nets are loaded into the agents via Goal Net Loader. As a result, the multi-agent system has been successfully developed.

3.5 Agent-based E-Learning Systems

An agent based e-learning system has been developed based on the e-learning case described in this paper. Experiments have been conducted for evaluating the system. Agents in the e-learning system developed based on the proposed GO methodology have both goal autonomy and behavior autonomy. In the e-learning system, the course delivery agent uses a Bayesian network based action selection mechanism to select suitable learning objects based on the learner's skills. There are two cases for consideration here: 1) the learner has already learned the learning object; 2) the learner has working experience related to the content of the learning object which means the learner has accu-

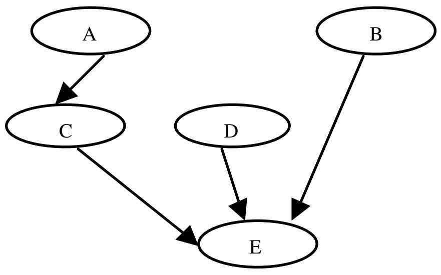


Fig. 6 The Bayesian network for learning object selection.

Table 1 The node definitions of the Bayesian network.

Node	Definition
A	used programming language
B	learned Java
C	used Java
D	duration that Java was used
E	learning object

Table 2 The probabilities between the nodes of the Bayesian network.

Node	A=y	B=y	C=y	D=1 yr	E=n
A=y	0.5				
B=y		0.5			
C=y	0.8		0.5		
D=1 yr				0.9	
E=n		1	0.5	0.5	0.5

mulated some skills about the content of the learning object through his working experience. For example, a learner, who has not taken a course about Java, can possess the skill about Java if he has worked in a project using Java for a certain period. So whether he needs to take the learning object about Java is decided by how long he has used Java for his work and the results of the pre-assessment. We assume that the longer he has used Java, the higher the probability that he possesses the required skill. In this way, Bayesian networks can be used to decide whether a learner needs to learn the Java learning object or not. So, in the transition, if the probability of having the skill that is related to the learning object is higher than a pre-set threshold, the learning object will not be selected. Otherwise, the learning object will be selected.

Figure 6 gives an example Bayesian network for the learning object selection. Table 1 lists the definitions of the nodes.

Table 2 lists a setting of the probabilities between the nodes. In Table 2, ”y” indicates the value ”yes” while ”n” indicates ”no”.

In this experiment, suppose the threshold for the probability not to select the learning object is 0.5, and a learner has working experience in Java programming, if he has worked for more than one year, the probability is $0.8*0.5 + 0.5 = 0.9 > 0.5$, which indicates that this learning object is not selected; if he has worked for less than one year, the probability is $0.8 * 0.5 = 0.4 < 0.5$, which means that the learning object should be selected. In contrast, if the learner has learned the learning object, the learning object is not selected because the probability is $1 > 0.5$.

Besides the above described e-learning system, MADE has been used in different application domains. Evaluation has been conducted for the agent development with and without MADE. Compared to the traditional agent-based system development in which developers encounter great difficulties in developing intelligent agents, GO methodology with MADE presents a user-friendly and easy-to-use approach to intelligent agent development.

Table 3 Evaluation of different agent development methodologies.

Evaluation Criteria	**Gaia**	**MaSE**	**Tropos**	**Goal Net**
Goal Oriented	Role based	Role based	Goal based	Goal based
Full Development Cycle	No	Yes	Yes	Yes
Toolkits	No	Yes	Yes	Yes
Developer Friendliness	Low	Mediate	Mediate	High

4. Related Work

To date there are several kinds of major methodologies for agent system development, such as Gaia [3], [6], MaSE [4], [5], and Tropos [7], etc. These methodologies are designed from different perspectives. In this section, we evaluate our proposed methodology against these popular agent development methodologies from the view of software engineering. Following are the evaluation factors in use:

- *Goal Oriented*: measures whether the methodology models the agent system in the goal oriented manner or not.
- *System Development Cycle*: measures whether the methodology supports the whole software development cycle from initial system requirement to the final implementation.
- *Toolkit*: measures whether the methodology has proposed some toolkit to assist the development of the agent system.
- *Developer Friendliness*: measures whether the methodology is user friendly to the developers, or whether it is easy to use.

Table 3 shows the results of the simple evaluation on different agent system development methodologies. As it is seen, only Tropos and the proposed methodology are goal oriented. However, Goal Net models the temporal relationships between goals and supports dynamic goal selection and action selection. Moreover, goal nets can be directly loaded into agents as the mental models of the agents. Therefore intelligent agent development becomes easier. In addition, the proposed methodology supports the full software development cycle. The MADE toolkit is highly user friendly and it is able to improve developers' efficiency significantly.

5. Conclusion

In this paper, we have presented a goal-oriented methodology for developing agent systems. This methodology is based on the Goal Net model, which models agent with not only behavior autonomy but also goal autonomy. The MADE toolkit, which provides a great assistance to developers in designing and implementing agent system, is also illustrated. A comparison with different existing agent methodologies from the software engineering perspective shows that the proposed methodology is a practical, easy-to-use and efficient methodology for agent system development.

References

[1] H. Nwana, and D. Ndumu, A Perspective on Software Agents Research, Knowledge Engineering Review, Vol. 14, No.2, pp. 1-18, 1999.

[2] Z. Q. Shen, Goal-oriented Modeling for Intelligent Agents and their Applications, Ph.D. Thesis, Nanyang Technological University, Singapore, 2003.

[3] F. Zambonelli, N. R. Jennings and M. J. Wooldridge, Developing multiagent systems: the Gaia Methodology, ACM Trans on Software Engineering and Methodology, Vol.12, No. 3, pp. 317-370, 2003.

[4] S. A. DeLoach, Systems Engineering A Methodology and Language for Designing Agent Systems, Proc. of Agent Oriented Information Systems, pp. 45-57, 1999.

[5] S. A. DeLoach, Analysis using MaSE and agentTool, Proc. of Midwest Artificial Intelligence and Cognitive Science Conference, 2001.

[6] M. J. Wooldridge, N. R. Jennings and D. Kinny, The Gaia methodology for agent-oriented analysis and design, Autonomous Agents and Multi-Agent Systems, 3, pp. 285-312, 2000.

[7] P. Bresciani , P. Giorgini , F. Giunchiglia , J. Mylopoulos and A. Perini , Tropos: An Agent-Oriented Software Development Methodology, Journal of Autonomous Agent and MultiAgent Systems, 8 (3), pp. 203-236, 2004.

[8] F. Bellifemine, A. Poggi and G. Rimassa, JADE: a FIPA2000 compliant agent development environment, in Proceedings of the fifth international conference on Autonomous agents, pp. 216 - 217, Montreal, Quebec, Canada, 2001.

[9] A. Dardenne, A. van Lamsweerde and S. Fickas, Goal-Directed Requirements Acquisition, Science of Computer Programming, Vol. 20, North Holland, pp. 3-50, 1993.

[10] A. Anton, Goal-Based Requirements Analysis, Second International Conference on Requirements Engineering, Los Alamitos, California: IEEE Computer Society Press, pp. 136-144, 1996.

[11] A. Anton, Goal Identification and Refinement in the Specification of Software-Based Information Systems, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, Georgia, June 1997.

[12] P. Massonet, Y. Deville and C. Nve, From AOSE methodology to agent implementation, AAMAS 2002, pp. 27-34, 2002.

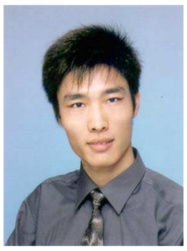
[13] Z. Q. Shen, R. Gay, C. Y. Miao and X. H. Tao, Goal Oriented Modeling for Intelligent Software Agents, in Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04), Beijing, China, September, 20-24, 2004.

[14] Foundation for Intelligent Physical Agents, FIPA Agent Management Specification, http://www.fipa.org/specs/fipa00023/, June 2002.

Zhiqi Shen Dr. Zhiqi Shen obtained his BSc in Computer Science in Peking University, China, and PhD in Information Communication Institute, Nanyang Technological University (NTU), Singapore. His research interests include goal-oriented modeling, intelligent software agent, software engineering, semantic web/grid, sensor network and their applications.

Chunyan Miao Dr. Chunyan Miao received her PhD from School of Computer Engineering, Nanyang Technological University (NTU), Singapore. She is currently an Assistant Professor in the same school. Her major research interest includes machine learning, intelligent software agent, agent mediated semantic web/grid, and agent oriented software engineering.

Robert Gay Professor Robert Gay obtained his PhD in Electronics Engineering from the University of Sheffield in 1970. He is currently Director of the Managed Computing Competency Centre and the Director for Research IT Resources at NTU. His current research interests and expertise include Semantic Grid, Knowledge Based Systems, E-learning and Integrated Manufacturing Systems and Services.



Dongtao Li Dongtao Li is currently a postgraduate research student in School of Computer Engineering, Nanyang Technological University, Singapore. His main research interests include Software Agent, Multi-Agent System, Agent-Oriented Software Engineering.