Intelligent Software Agent Design Tool Using Goal Net Methodology

Han Yu, Zhiqi Shen and Chunyan Miao
Nanyang Technological University
yuhan@pmail.ntu.edu.sg, zqshen@ntu.edu.sg, ascymiao@ntu.edu.sg

Abstract
Intelligent agent is a fast emerging technology and has wide range of applications. Although there are several tools for agent development, there is few design tool to assist the conversion from paper based agent mental state design to effective representation of them in abstract data structures which can be used by the agent management system to create intelligent software agents. This paper proposes the Goal Net Designer which is an Integrated Development Environment (IDE) for modeling agent behavior based on Goal Net model, a goal orient methodology. It provides a way for the users to simplify the various stages of the design process and automatically generate design data which can be used by the Multi-Agent Development Environment (MADE) to automatically create intelligent agents. The system reduces the level of skills required for developing agent augmented application to such an extent that users with little knowledge in intelligent software agent technology can easily add intelligent agents into their applications and save time and cost involved in the development process.

1. Introduction
The goal oriented model most closely resembles how people, in their day to day activities, make decision to direct their actions. It has great potential to be understood easily by people with little expertise in intelligent software agent technology. Nowadays, most of the available intelligent agent design tools are targeted at users who are experts or at least highly skilled in the field of intelligent agent to design a multi-agent system or agent mental states and even more highly competent in programming to implement the design. Therefore, the aim of this research is to provide a tool for less technically inclined people (game plot writers, system managers etc.) to enable them to actively participate in the design of multi-agent systems.

The methodology based on Goal Net model is a goal oriented agent development methodology that covers the entire life cycle of the agent system development from requirement analysis, architectural design and detailed design to implementation [1]. It has a practical implementation namely the Multi-Agent Development Environment (MADE) which consists of two main parts: The Goal Net Designer and the MADE Runtime. Out of the four main phases, Goal Net Designer covers the requirement analysis and the detailed design. The fourth phase is supported by the MADE Runtime which depends on designed goal nets and data acquired with the Goal Net Designer.

In the following sections, we will give an overview of the architecture of MADE, discuss important features of the Goal Net Designer tool and a step-by-step demonstration of how to design and implement a simple agent using the tool.

2. Related Work
Currently, there are a number of approaches for designing multi-agent systems. One of the most popular tools for agent development is the Java Agent DEvelopment Framework (JADE). It provides APIs for developer to implement their agent designs in Java programming language [2]. However, it is not a development tool for mental state design. In order to create an agent in JADE, one has to be familiar with both the Java programming language as well as the agent oriented programming paradigm.

Another approach for designing multi-agent systems is to sketch the design on paper and feed it to a computer with capable software. In most cases, the software is platform and language dependent. Although the SketchiXML project offers a solution to this problem [3], it has nevertheless some inherent drawbacks by using this approach. This approach does not enable the users to specify internal properties of each design item during the sketching phase. The skill
level required to make use of this design approach is high and the implementation is still a long way to go from the design.

The Prometheus Design Tool (PDT) proposed by RMIT University, Australia, is a very popular agent design tool. It provides a wide range of support for the various stages of multi-agent system design using the Prometheus methodology. It has been integrated with Eclipse to provide a more closely coupled implementation process [4]. However, the data acquired at design time serves only as a reference to the implementation process (which still has to be accomplished manually by writing program code). The correlation between the software agent implementation and agent data collected from PDT is not set up automatically.

3. Overview of MADE

3.1 Goal Net Model

The Goal Net model and the agent development methodology was proposed by Dr. Shen [5]. A goal net is a composite goal hierarchy which is composed of goals (alternatively known as states) and transitions. The goals are used to represent the states that an agent needs to go through in order to achieve its final goal. The transitions connect one goal to another 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 which might be performed by the agent in order to transit from the input goal to the output goal. 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. A Goal Net model defines behaviors of an agent that executes it. Each agent has at least one goal net. A complex goal net can be split to many goal nets. Therefore a multi-agent system can be formed.

3.2 MADE Architecture

As shown in Fig 1, the Multi-Agent Development Environment (MADE) provides a framework in which multi-agent systems can be designed, implemented and run. Of these three stages, users are mainly involved in the design process. The implementation of agents using MADE still requires certain amount of program code to support functions required by a specific application. This is needed when the agents are set to run under a customized environment (e.g. an online virtual 3D community). But the coding involves no agent concept. Other than this part which requires the participation of technically inclined programmers, the agent implementation and execution of the multi-agent system according to the design have been automated by MADE.

<table>
<thead>
<tr>
<th>Goal Net Designer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MADE Runtime</td>
</tr>
<tr>
<td>Agent Management Platform</td>
</tr>
<tr>
<td>Agent Creator</td>
</tr>
<tr>
<td>Goal Net Enabled Agent Framework</td>
</tr>
</tbody>
</table>

The designers/users (usually people with limited technical knowledge), makes use of a GUI tool – the Goal Net Designer- to draw a goal net based on the dependencies among the goals and transitions. This process is straightforward as it models how human logic works, so only minimal training is required to get familiar with the features of the tool in order to complete the design. After the Goal Net design, the designed goal net and the design data are saved in a Database. Goal Net Designer is an important component of MADE. Fig 2 shows the architecture of the Goal Net Designer.
Agents are created and run within the MADE Runtime which is essentially a virtual machine that interprets the goal net and the design data. Whenever a reference to a routine defined in a transition which is specific to the customized environment of the application is encountered, the MADE runtime would redirect the call to the DLL files provided by the developers. The function defined in the DLL file would be executed in the customized environment.

4. The Goal Net Designer in Action

4.1 Overview of the Goal Net Designer

The Goal Net Designer provides an easy to use drawing canvas for the users to complete architectural design of the multi-agent system with as little effort spent on drawing and layout as possible. Thus, the entities involved in a goal net – atomic states, composite states, simple transitions, reasoning transitions, directed arcs and parallel arcs etc, are all provided as standard components which can be dragged and dropped and moved about on the canvas.

It contains mechanisms to facilitate detailed design of the agent mental states by allowing users to associate goal selection algorithms in the form of functions with the target goals or transitions. The functions used in a transition can be grouped into tasks so they can be executed one by one in predefined sequence just like batch processing. This enables a wide range of complex actions to be accomplished by reusing a limited set of basic functions arranged into different tasks. These application specific functions are provided in the form of DLL files.

To facilitate collaborative development with developers stationed in diverse physical locations, the Goal Net Designer adopted the client-server architecture. A centralized database is used to store the designed goal nets and the design data. Messaging services are also provided for users to communicate with each other among themselves during the design time.

4.2 Designing an Agent with Goal Net Designer

To demonstrate how the tool works, this section shows how a simple agent comes to life from the Goal Net design in step-by-step. For the purpose of illustration, the agent should say “Hello”, walk to a new location and say “Goodbye” before stopping. The test bed used for this demonstration is the Singapore River City project which is based on the Active World online 3D environment so that the actual activities of the resulting agent can be directly observed.

![Goal Net for HelloAgent](image)

The first step is the design of the goal net of the agent. As shown in Fig 3, the goal net consists of four distinct states (circles) each with a name that is self-explanatory. The archs (arrows) indicate the flow of logic from one state to the next. To go from state to state, a transition (round-corner rectangle) must be passed through. It is in these transitions that the actions the agent must perform are associated with their actual implementation functions. The composite state (circle 168) acts as a high level representation of the entire goal net. It can be reused by implanting itself into another goal net. The numbers overlaying on the graphical entities are their respective identification numbers. Each is unique within its own type but not across type boundaries (a state and a transition might have the same ID number).

The next step is to specify which function should be associated to a transition. A task is first created for a transition as a container to accommodate functions. Multiple tasks can be created for each transition, but a transition must have at least one task. After this, functions can be dragged from a function list and dropped on the desired transition on the canvas. Some functions require arguments to be passed to them. In our case, the *say()* function has to know “what” the agent has to say. By right clicking on the transition, a form shows up and enables the users to add in a number of parameters according to the function signature. To populate the function list, the user should upload the DLL file containing all the functions needed for this goal net into a designated *Functions* folder on the server computer. The server program periodically polls this folder for new DLL files, if found, the function signatures of the new DLL files will be extracted into the central database and made available to all users. All active clients will be notified of the new change via callback functions.
Each state and transition has a set of properties like names, descriptions which can be edited to facilitate a better understanding of the design by other users. Apart from these informational properties, other properties like cost of going through a transition can be specified by the users which will be modified during runtime to facilitate the execution of the default goal selection algorithm provided by the MADE runtime. Fig 4 illustrates a property pan with default values assigned to all fields. The designed goal net is saved into the database with a name after the design finishes.

![Fig 4. Property Grid for a Transition](image)

The last step is to create an agent and run it in the MADE runtime. The name of a goal net (which is checked for collision during design time and guaranteed to be unique) is passed to the agent as the argument. MADE runtime will create an agent, load the goal net from the database and interpret the goal net from the starting state. Whenever a defined function is encountered, it is invoked by pushing the function name and parameters to the system stack. In this way, in our example, the agent which manifests itself as a 3D avatar in the Singapore River City project test bed will execute the actions as defined in the goal net. Fig 5 shows a screen shot of the agent controlled 3D characters in our virtual environment.

![Fig 5. Agent Controlled 3D Avatars Built with MADE](image)

5. Conclusions and Future Work

With the Goal Net Designer and MADE Runtime, users can be freed, to a great extent, of the concern of implementing their agent designs which is a process normally requiring in-depth knowledge of the intelligent agent technology and programming. With the Goal Net methodology, developers only need to program the functions in traditional way and design goal nets using the Goal Net Designer. The tool has been used in our research projects and it has been demonstrated to considerably reduce the time and effort required to design and implement multi-agent systems.

In the further development of the Goal Net Designer, the Goal Net split and reuse will be investigated so that a complex goal net can be designed using this tool. To enhance user experience and collaboration of team members in a virtual community in the design of multi-agent systems, a web version of the entire Goal Net design system will be set up. Users will be able to use it as an embedded web component which can be easily updated/ upgraded without involvement of users. The design process will be conducted in the virtual community so that users can have more interaction with each other.

6. References


